

OCSS7 Installation and Administration Guide

TAS-072-Issue 1.1.0-Release 1

February 2018

metaswitch

Notices

Copyright © 2017 Metaswitch Networks. All rights reserved.

This manual is issued on a controlled basis to a specific person on the understanding that no part of the Metaswitch Networks product code or documentation (including this manual) will be copied or distributed without prior agreement in writing from Metaswitch Networks.

Metaswitch Networks reserves the right to, without notice, modify or revise all or part of this document and/or change product features or specifications and shall not be responsible for any loss, cost, or damage, including consequential damage, caused by reliance on these materials.

Metaswitch and the Metaswitch logo are trademarks of Metaswitch Networks. Other brands and products referenced herein are the trademarks or registered trademarks of their respective holders.

Contents

1 OCSS7 Installation and Administration Guide.....	13
1.1 Topics.....	13
2 About OCSS7.....	14
2.1 What is OCSS7?.....	14
2.2 Main features.....	14
2.3 Supported specifications.....	14
2.4 Architecture.....	15
2.4.1 SGC Subsystems overview.....	17
2.4.2 SGC Deployment model.....	17
2.5 Management tools.....	17
3 Quick Start.....	19
3.1 Introduction.....	19
3.2 The Plan.....	20
3.3 SGC installation.....	20
3.4 SGC cluster membership configuration.....	22
3.5 Starting the clusters.....	23
3.6 Connect the management console.....	25
3.7 General configuration.....	26
3.8 M3UA configuration.....	29
3.9 M3UA state inspection.....	36
3.10 SCCP configuration.....	39
3.11 SCCP state inspection.....	42

3.12 Scenario Simulator installation.....	44
3.13 Scenario Simulator configuration.....	47
3.14 Test the network.....	48
4 Installing the SGC.....	52
4.1 Checking prerequisites.....	52
4.2 Configuring network features.....	52
4.3 User process tuning.....	53
4.4 SCTP tuning.....	54
4.5 SGC Installation.....	55
4.5.1 Unpack and configure.....	55
4.5.2 Creating additional nodes.....	56
4.5.3 Layout of the SGC installation.....	57
4.6 Running the SGC.....	57
4.6.1 SGC operations.....	57
4.6.2 Configuring SGC_HOME/config/sgcenv.....	59
4.7 Installing SGC as a service.....	62
5 Network Architecture Planning.....	63
5.1 Network planning.....	63
5.2 SGC Stack network communication overview.....	63
5.2.1 SGC directly managed connections.....	64
5.2.2 Hazelcast managed connections.....	66
5.2.3 Inter-node message transfer.....	67
Outgoing message inter-node transfer.....	67
Incoming message inter-node transfer.....	67
Avoiding communication switch overhead.....	68

5.3 SGC cluster membership and split-brain scenario.....	69
5.4 Recommended physical deployment model.....	70
6 Securing the SGC JMX management connection with SSL/TLS.....	74
6.1 Default configuration of the JMX management connection.....	74
6.2 Securing the JMX management connection with SSL/TLS.....	74
6.2.1 SGC stack secure configuration.....	74
6.2.2 Example client configuration for a JMX management secure connection.....	75
Configuring from the command line.....	75
Configuring with a generic JMX management tool.....	75
6.3 SGC stack JMX configuration properties.....	76
6.3.1 Properties configurable using the sgcnv configuration file.....	76
6.3.2 Example JMX_SECURE_CFG_FILE properties file.....	77
6.4 SGC stack JMX connector configuration details.....	78
7 Configuring the SS7 SGC Stack.....	81
7.1 Configuration data.....	81
7.1.1 Static SGC configuration.....	81
Static SGC instance configuration.....	81
7.1.2 Hazelcast cluster configuration.....	88
7.1.3 Logging configuration.....	90
7.2 Managed SGC cluster configuration.....	90
7.2.1 Configuration objects.....	90
7.2.2 Common configuration object attributes.....	91
8 General Configuration.....	93
8.1 node.....	93
8.2 parameters.....	95

9 M3UA Configuration.....	97
9.1 Application Server and routes.....	97
9.1.1 as.....	97
9.1.2 route.....	98
9.1.3 dpc.....	99
9.1.4 as-precond.....	100
9.1.5 as-connection.....	100
9.2 Listening for and establishing SCTP associations.....	101
9.2.1 local-endpoint.....	101
9.2.2 local-endpoint-ip.....	103
9.2.3 connection.....	104
9.2.4 conn-ip.....	105
10 SCCP Configuration.....	107
10.1 Global title translation.....	107
10.2 Incoming GT translation.....	107
10.2.1 inbound-gtt.....	107
10.3 Outgoing GT translation.....	109
10.3.1 outbound-gt.....	109
10.3.2 outbound-gtt.....	110
10.3.3 replace-gt.....	111
10.4 Concerned Point Codes.....	112
10.4.1 cpc.....	112
10.5 Load balancing.....	113
11 SNMP Configuration.....	114
11.1 Interoperability with SNMP-aware management clients.....	114

11.2 SNMP configuration.....	114
11.2.1 snmp-node.....	114
11.2.2 target-address.....	115
11.2.3 usm-user.....	117
11.3 SGC Stack MIB definitions.....	118
11.3.1 SNMP managed objects.....	118
Statistics managed objects.....	118
Alarms managed objects.....	119
11.3.2 SNMP notifications.....	120
12 Configuration Procedure.....	122
12.1 General configuration.....	122
12.2 SCCP configuration.....	123
12.2.1 Outgoing GT.....	123
12.2.2 Incoming GT.....	123
12.2.3 Concerned Point Code.....	124
12.3 M3UA configuration.....	124
13 Configuration Subsystem Details.....	126
13.1 Stack configuration and cluster joining.....	126
13.2 Cluster-join procedure.....	126
13.2.1 Version conflict reconciliation.....	127
13.3 Factory MBeans for configuration objects.....	127
13.3.1 Configuration MBean naming.....	128
14 Operational State and Instance Management.....	129
14.1 Operational state.....	129
14.2 Static SGC instance configuration.....	129

15 Alarms.....	131
15.1 What are SS7 SGC alarms?.....	131
15.2 Active alarms and event history.....	131
15.3 Generic alarm attributes.....	132
15.4 Alarm types.....	133
15.4.1 General alarms.....	133
commswitchbindfailure.....	133
distributedDataInconsistency.....	133
nodefailure.....	134
poolCongestion.....	135
poolExhaustion.....	135
workgroupCongestion.....	136
15.4.2 M3UA alarms.....	137
asDown.....	137
asConnDown.....	137
associationCongested.....	138
associationDown.....	138
dpcRestricted.....	139
dpcUnavailable.....	139
15.4.3 SCCP alarms.....	140
sccpLocalSsnProhibited.....	140
sccpRemoteNodeCongestion.....	141
sccpRemoteNodeNotAvailable.....	141
sccpRemoteSsnProhibited.....	142
16 Notifications.....	143

16.1 What are SS7 SGC notifications?.....	143
16.2 How are notifications different from alarms?.....	143
16.3 Generic notification attributes.....	144
16.4 Notification types.....	144
16.4.1 MTP decode errors.....	145
16.4.2 SCCP decode errors.....	146
16.4.3 TCAP decode errors.....	146
16.4.4 TCAP stack register.....	147
16.4.5 TCAP stack unregister.....	148
17 Statistics.....	149
17.1 What are SS7 SGC statistics?.....	149
17.2 Statistics.....	150
17.2.1 M3UA.....	150
AsInfo.....	150
AssociationInfo.....	151
DpcInfo.....	152
17.2.2 SCCP.....	153
LocalSsnInfo.....	153
OgtInfo.....	154
PcInfo.....	155
RemoteSsnInfo.....	156
17.2.3 TCAP.....	156
TcapConnInfo.....	156
17.2.4 TOP.....	157
HealthInfo.....	157

18 Logging.....	159
18.1 About the Logging subsystem.....	159
18.2 Logger names, levels, appenders.....	159
18.2.1 Logger names.....	159
18.2.2 Log levels.....	160
18.2.3 Log appenders.....	160
Rolling file appenders.....	161
18.3 Default logging configuration.....	161
18.3.1 Default appenders.....	161
19 JMX MBean Naming Structure.....	163
19.1 Instance Management MBean.....	163
19.1.1 Properties attribute.....	163
19.1.2 Operations.....	164
19.2 Alarm MBean naming structure.....	164
19.3 Alarm raised and cleared notifications.....	166
19.4 Active alarms and alarm history.....	166
19.4.1 Alarm Table.....	167
19.4.2 Alarm History.....	168
19.5 Notification MBean naming structure.....	168
19.6 Statistics MBean naming structure.....	169
19.6.1 Subsystem statistics.....	169
19.6.2 Processing object-specific statistics.....	171
20 Command-Line Management Console.....	173
20.1 What is the SGC Stack Command-Line Console?.....	173
20.2 Installation and requirements.....	173

20.3 Working with the SGC CLI.....	174
20.3.1 Enabling secured JMX server connection.....	175
20.3.2 Basic command format.....	175
20.3.3 MML syntax auto-completing.....	176
20.3.4 Help mode.....	177
20.3.5 Interactive mode.....	179
20.3.6 Command result truncation.....	179
21 Supported CLI Operations.....	181
21.1 SGC CLI commands.....	181
21.1.1 Management of SGC Stack processing objects.....	181
21.1.2 Alarms and event history.....	184
Displaying all active alarms (no filtering criteria specified):.....	184
Displaying all active alarms with filters:.....	185
Displaying all active alarms with filters and column parameters:.....	185
21.1.3 Clearing an active alarm:.....	186
21.1.4 Displaying all registered alarms:.....	186
21.2 Statistics (Info).....	187
21.2.1 Displaying statistic without filters:.....	188
21.2.2 Displaying asinfo statistics with filters on the nodeId:.....	188
21.3 Export / Import.....	188
22 SGC TCAP Stack.....	191
22.1 What is the SGC TCAP Stack ?.....	191
22.2 TCAP Stack and SGC Stack cooperation.....	191
22.2.1 'ocss7.sgcs' Connection Method.....	191
TCAP Stack registration process.....	191

Load balancing TCAP stack data.....	192
Prefix Migration.....	192
22.2.2 Legacy 'ocss7.urlList' Connection Method.....	194
TCAP Stack registration process.....	196
Data transfer connection failover.....	197
22.3 TCAP Stack configuration.....	199
22.4 SGC-TCAP stack protocols.....	203
22.4.1 v2.0.0.....	203
Handshake.....	203
Heartbeat.....	204
22.4.2 v1.0.0.....	204
Handshake.....	204
Data Connection Heartbeat Mechanism.....	204
23 Upgrading the SGC and TCAP stack.....	206
23.1 Upgrading from OCSS7 1.0.1.x to OCSS7 1.1.0.x.....	206
23.1.1 Notable Changes.....	206
23.1.2 Online Upgrade Using STP Redirection.....	206
Prerequisites.....	206
Upgrade process.....	207
24 Glossary.....	208

1 OCSS7 Installation and Administration Guide

This guide describes how to deploy, manage and maintain OCSS7 - OpenCloud's SS7 Stack.

This document assumes a working knowledge of SS7 and SIGTRAN, as well as some familiarity with the administration of Linux systems and Java applications.

1.1 Topics

This document includes the following topics:

About OCSS7	An overview of OCSS7, including features and architecture
Quick Start	A step-by-step setup guide from packages through a test dialog, ignoring production-grade details
Installing the SGC	SGC installation in detail, including production-grade features and tuning
Configuring the OCSS7 SGC	Configuration options for startup and runtime operations
Operational State and Instance Management	Monitoring OCC7's operational state
Command-Line Management Console	Driving OCCS7 from the command line
SGC TCAP Stack	Use of OCSS7 with OpenCloud's CGIN Connectivity Pack
Upgrading the SGC and TCAP stack	Upgrading the SGC and the CGIN TCAP stack

Other documentation for OCSS7 can be found on the [OCSS7 product page](#) .

2 About OCSS7

2.1 What is OCSS7?

OCSS7 provides SS7 network connectivity to OpenCloud products, including the Rhino CGIN RA and the OpenCloud Scenario Simulator. It does this by implementing a set of SS7 and related SIGTRAN protocols. The current version supports M3UA, ITU-T SCCP, and ITU-T TCAP, and can act as an M3UA Application Server connected either to an M3UA Signalling Gateway or to another Application Server as an IP Server Process.

2.2 Main features

OCSS7 offers the following main features:

- Provides IN network connectivity to OpenCloud's CGIN and Scenario Simulator
- SIGTRAN M3UA communication with Signalling Gateways and/or IPSP peers
- Cluster deployment — SGC may be deployed in a cluster where multiple SGC instances cooperate and represent a single Signalling Point Code.

2.3 Supported specifications

Specification	Comment
RFC 4666	M3UA (ASP and IPSP functionality)
ITU-T Q.711-Q.714	SCCP (excluding segmentation of LUDT)
ITU-T Q.771-Q.774	TCAP (constrained to features supported by the TCAP SPI interface provided by the CGIN RA)

More details on supported RFCs, see the [OCSS7 Compliance Matrix](#) .

2.4 Architecture

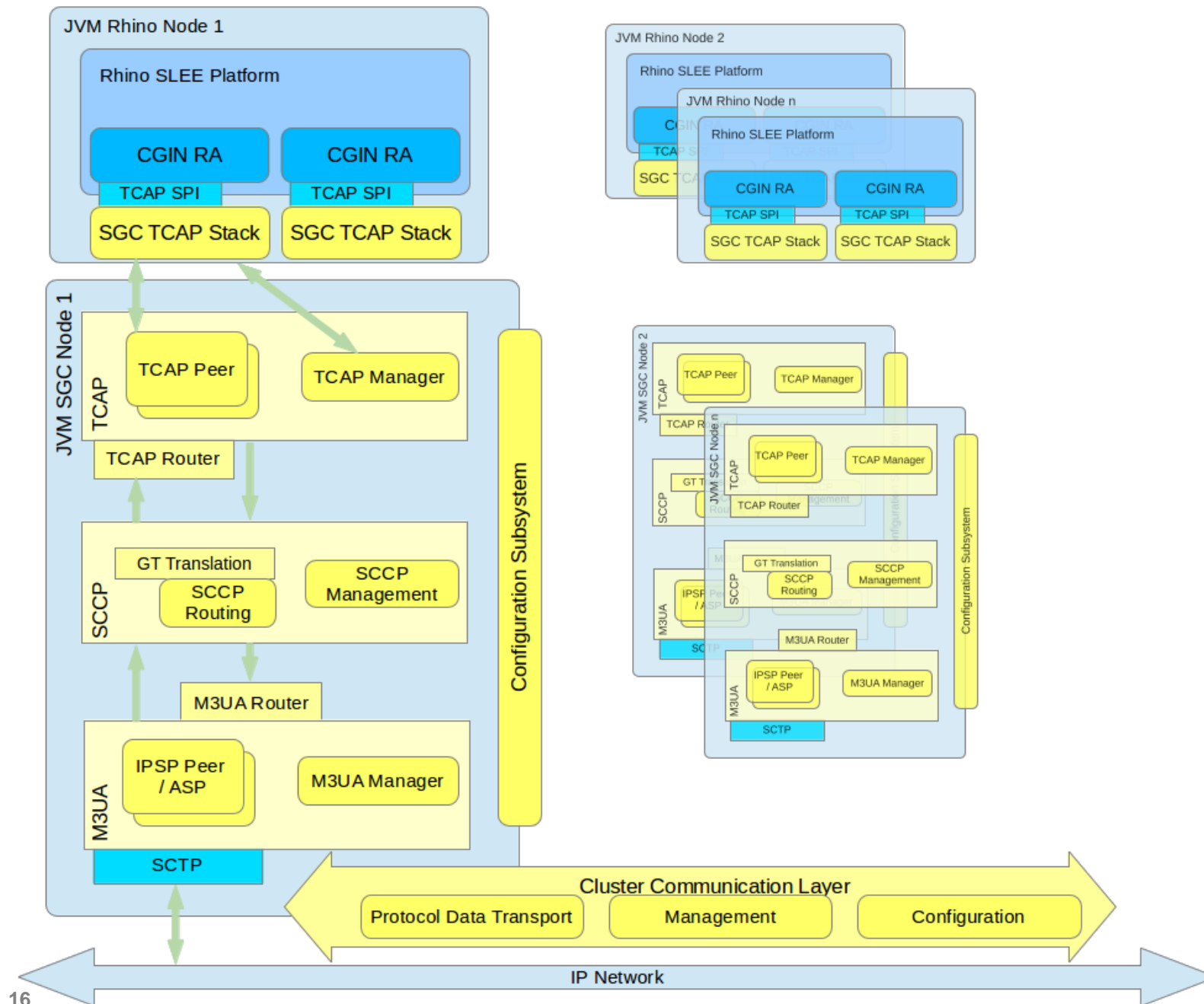
OCSS7 is composed of two main user-visible components:

- the Signalling Gateway Client (SGC) TCAP Stack (frontend), automatically deployed as part of the Rhino CGIN RA, and
- the SGC (backend), deployed as one or more separate processes.

After initialization, the TCAP Stack (frontend) registers with a preconfigured SSN at the SGC instance, and allows the CGIN RA to accept and initiate TCAP dialogs. Multiple TCAP Stack (frontend) instances representing the same SSN can be connected to the SGC instance (backend); in this case all incoming traffic will be load balanced among them.

The SGC provides IN connectivity for exactly one Point Code. Typically, the SGC is deployed as a cluster of at least two instances to meet standard carrier-grade, high-availability requirements. This manual primarily concerns itself with the OCSS7 SGC, as the TCAP stack component within CGIN RA is managed as part of CGIN.

Below is a high-level diagram followed by descriptions of the main components of the OCSS7 stack. The OCSS7 stack components are yellow . Components that are provided by the Rhino platform, CGIN, or the operating system environment are blue .



2.4.1 SGC Subsystems overview

All SGC nodes in the cluster represent a single logical system with a single PC (SS7 Point Code). Each SGC cluster node instantiates all stack layers and subsystems, which coordinate with each other to provide a single logical view of the cluster.

The major SS7 SGC subsystems are:

- Configuration Subsystem — plays a central role in managing the life cycle of all processing objects in the SGC Stack. The Configuration Subsystem is distributed, so an object on any cluster node may be managed from every other cluster node. Configuration is exposed through a set of JMX MBeans which allow manipulation of the underlying configuration objects
- TCAP Layer — manages routing and load balancing of TCAP messages to appropriate Rhino CGIN RAs (through registered TCAP Stacks)
- SCCP Layer — responsible for routing SCCP messages, [GT](#) on page translation, and managing internal and external subsystem availability states
- M3UA Layer — establishes and manages the [SG](#) on page , [IPSP](#) on page connectivity, and [AS](#) on page state.

Internally, all layers use Cluster Communication subsystems to distribute management state and configuration data, to provide a single logical view of the entire cluster. Irrespective of which cluster node receives or originates an SS7 message, that message is transported to the appropriate cluster node for processing (by the Rhino CGIN RA) or for further sending using one of the nodes' established SCTP associations.

2.4.2 SGC Deployment model

The SGC cluster physical deployment model is independent from Rhino cluster deployment. That is, SGC nodes can be deployed on separate hosts than Rhino nodes OR share the same machines.



See also [Installing the SGC](#) on page 52

2.5 Management tools

The SGC installation package provides a [CLI](#) on page management console that exposes a set of [CRUD](#) on page commands that are used to configure the SGC cluster and observe its runtime state. SGC cluster configuration and runtime state can be managed from each and

every node; there is no need to connect separately to each node. As the SGC exposes a standard Java JMX management interface, users are in no way constrained to provided tools and are free to create custom management clients to serve their particular needs.

The SGC also provides an [SNMP](#) on page agent, exposing SGC-gathered statistics and Alarm notifications (Traps) through the SNMP protocol.

The TCAP stack component installed as part of the Rhino CGIN RA is managed using the usual Rhino and CGIN RA management facilities.

3 Quick Start

This section provides a step-by-step walk through basic OCSS7 setup from unpacking software packages through running test traffic. The end result is a functional OCSS7 network suitable for basic testing. For production installations and installation reference material please see [Installing the SGC](#) on page 52 .

3.1 Introduction

In this walk-through we will be:

- setting up two OCSS7 clusters, each with a single SGC node, and
- running one of the example IN scenarios through the network using the OpenCloud Scenario Simulator.

To complete this walk-through you will need the:

- OCSS7 package,
- [OpenCloud Scenario Simulator 2.3.0](#)
- [IN Scenario Pack 1.5.3](#) or higher for the Scenario Simulator, and
- a [compatible Java Virtual Machine](#) .

These instructions should be followed on a test system which:

- runs Linux,
- has [SCTP support](#) , and
- is unlikely to be hampered by local firewall or other security restrictions.

Finally, you will need to make sure that the `JAVA_HOME` environment variable is set to the location of your Oracle Java 7 JDK installation.

3.2 The Plan

We will set up two clusters, each with a single node, both running on our single test system. At the M3UA level:

- cluster 1 will use Point Code 1
- cluster 2 will use Point Code 2
- there will be one Application Server (AS) with routing context 2
- there will be one SCTP association between the two nodes

We will test the network using two OpenCloud Scenario Simulators:

- Simulator 1 will:
 - connect to cluster 1
 - use SSN 101
 - use GT 1234
- Simulator 2 will:
 - connect to cluster 2
 - use SSN 102
 - use GT 4321

Routing between the two simulators will be via Global Title translation.

Once we think that the network is operational we will test it by running one of the example scenarios shipped with the IN Scenario Pack for the Scenario Simulator.

3.3 SGC installation

We will now install two SGC clusters, each containing one node:

- | | |
|---|-----------------------------|
| 1 | Unpack the SGC archive file |
|---|-----------------------------|


```
unzip ocss7-package-VERSION.zip
```

(replacing ocss7-package-VERSION.zip with the correct file name).

This creates the distribution directory, ocss7-X.X.X.X , in the current working directory.

Example:

```
$ unzip ocss7-package-1.1.0.0.zip Archive: ocss7-package-1.1.0.0.zip creating: ocss7-1.1.0
.0/ inflating: ocss7-1.1.0.0/CHANGELOG inflating: ocss7-1.1.0.0/README creating:
ocss7-1.1.0.0/config/ creating: ocss7-1.1.0.0/doc/ creating: ocss7-1.1.0.0/license/
creating: ocss7-1.1.0.0/logs/ creating: ocss7-1.1.0.0/var/ inflating: ocss7-1.1.0.0/
config/SGC.properties inflating: ocss7-1.1.0.0/config/SGC_bundle.properties.sample
inflating: ocss7-1.1.0.0/config/log4j.dtd inflating: ocss7-1.1.0.0/config/log4j.test.xml
inflating: ocss7-1.1.0.0/config/log4j.xml inflating: ocss7-1.1.0.0/config/sgcenv inflating:
ocss7-1.1.0.0/license/LICENSE.guava.txt inflating: ocss7-1.1.0.0/license/LICENSE.hazelc
ast.txt inflating: ocss7-1.1.0.0/license/LICENSE.log4j.txt inflating: ocss7-1.1.0.0/licens
e/LICENSE.netty.txt inflating: ocss7-1.1.0.0/license/LICENSE.protobuf.txt inflating:
ocss7-1.1.0.0/license/LICENSE.slf4j.txt inflating: ocss7-1.1.0.0/license/LICENSE.snmp
4j.txt creating: ocss7-1.1.0.0/bin/ inflating: ocss7-1.1.0.0/bin/generate-report.sh
inflating: ocss7-1.1.0.0/bin/sgc inflating: ocss7-1.1.0.0/bin/sgcd inflating: ocss7-1.1.0.0/
bin/sgckeygen inflating: ocss7-1.1.0.0/sgc.jar creating: ocss7-1.1.0.0/lib/ inflating:
ocss7-1.1.0.0/lib/apache-log4j-extras-1.2.17.jar inflating: ocss7-1.1.0.0/lib/guava-14.
0.1.jar inflating: ocss7-1.1.0.0/lib/hazelcast-2.6.5.jar inflating: ocss7-1.1.0.0/lib/jsr3
05-1.3.9.jar inflating: ocss7-1.1.0.0/lib/log4j-1.2.17.jar inflating: ocss7-1.1.0.0/lib/
netty-buffer-4.0.28.jar inflating: ocss7-1.1.0.0/lib/netty-codec-4.0.28.jar inflating:
ocss7-1.1.0.0/lib/netty-codec-http-4.0.28.jar inflating: ocss7-1.1.0.0/lib/netty-comm
on-4.0.28.jar inflating: ocss7-1.1.0.0/lib/netty-handler-4.0.28.jar inflating: ocss7-1.1.0
.0/lib/netty-transport-4.0.28.jar inflating: ocss7-1.1.0.0/lib/protobuf-java-2.3.0.jar
inflating: ocss7-1.1.0.0/lib/protobuf-library-2.3.0.1.jar inflating: ocss7-1.1.0.0/
lib/slf4j-api-1.7.7.jar inflating: ocss7-1.1.0.0/lib/slf4j-log4j12-1.7.7.jar inflating:
ocss7-1.1.0.0/lib/snmp4j-2.2.2.jar inflating: ocss7-1.1.0.0/lib/snmp4j-agent-2.0.1
0a.jar creating: ocss7-1.1.0.0/cli/ inflating: ocss7-1.1.0.0/cli/sgc-cli.sh creating:
ocss7-1.1.0.0/cli/conf/ creating: ocss7-1.1.0.0/cli/lib/ inflating: ocss7-1.1.0.0/cli/
conf/cli.properties inflating: ocss7-1.1.0.0/cli/conf/log4j.xml inflating: ocss7-1.1.0.0/
cli/lib/commons-cli-1.2.jar inflating: ocss7-1.1.0.0/cli/lib/commons-collections-3.2.1.ja
r inflating: ocss7-1.1.0.0/cli/lib/commons-lang-2.6.jar inflating: ocss7-1.1.0.0/cli/lib/
jline-1.0.jar inflating: ocss7-1.1.0.0/cli/lib/log4j-1.2.17.jar inflating: ocss7-1.1.0.0/
cli/lib/ocss7-cli.jar inflating: ocss7-1.1.0.0/cli/lib/ocss7-remote-1.1.0.0.jar inflating:
```

	<pre>ocss7-1.1.0.0/cli/lib/slf4j-api-1.7.7.jar inflating: ocss7-1.1.0.0/cli/lib/slf4j-log4j12-1.7.7.jar inflating: ocss7-1.1.0.0/cli/lib/velocity-1.7.jar inflating: ocss7-1.1.0.0/cli/sgc-cli.bat creating: ocss7-1.1.0.0/doc/mibs/ inflating: ocss7-1.1.0.0/doc/mibs/COMPUTARIS-MIB.txt inflating: ocss7-1.1.0.0/doc/mibs/CTS-SGC-MIB.txt inflating: ocss7-1.1.0.0/config/hazelcast.xml.sample</pre>
2	<p>Create SGC node PC1-1's installation</p> <pre>mv ocss7-X.X.X.X/ PC1-1</pre> <p>Example:</p> <pre>mv ocss7-1.1.0.0/ PC1-1</pre> <div>  <p>This creates the installation for our first SGC node. In this example we are following an SGC naming convention which lists the point code first and the node within the SGC cluster after a hyphen. During this walk-through the number after the hyphen will always be 1, but this convention provides space to expand if you wish to add additional nodes after completing the walk through.</p> </div>
3	<p>Create SGC node PC2-1's installation</p> <pre>cp -a PC1-1 PC2-1</pre>

We now have two SGC nodes with no configuration. The next step is to set up their cluster configuration.

3.4 SGC cluster membership configuration

We will now do the cluster membership configuration for our two SGC nodes/clusters, which for a single node cluster simply means setting the `ss7.instance` name. We shall set each `ss7.instance` name to match the name of the node's installation directory. Later on, during SS7 configuration, this instance name will be used to specify which node in the cluster certain configuration elements (such as SCTP endpoints) are associated with.

1	Give node PC1-1 its identity
----------	-------------------------------------

	<p>Edit the file <code>PC1-1/config/SGC.properties</code> and set <code>ss7.instance</code> to <code>PC1-1</code> . When you are done the file should look like this:</p> <pre># SGC instance node name ss7.instance=PC1-1 # Path to the Hazelcast config file hazelcast. config.file=config/hazelcast.xml # Default Hazelcast group name #hazelcast.group=abc123 #path where sgc data file should be stored sgc.data.dir=var</pre>
2	<p>Give node PC2-1 its identity</p> <p>Edit the file <code>PC2-1/config/SGC.properties</code> and set <code>ss7.instance</code> to <code>PC2-1</code> . When you are done the file should look like this:</p> <pre># SGC instance node name ss7.instance=PC2-1 # Path to the Hazelcast config file hazelcast. config.file=config/hazelcast.xml # Default Hazelcast group name #hazelcast.group=abc123 #path where sgc data file should be stored sgc.data.dir=var</pre>




For clusters with multiple nodes the `hazelcast.group` must be set, see [the installation reference](#) on page 55 . When left unconfigured, as we have done here, each node will create a unique group for itself, so we may be confident that our two nodes will not start up and attempt to cluster with each other.

If you wish to try adding a second node to one of the clusters after completing this walk-through you may wish to set the `hazelcast.group` to some value now. Any distinct values will do, for example, `PC1` and `PC2` would fit well with the naming convention used throughout this walk-through.

3.5 Starting the clusters

We will now start the two SGC clusters.

1	<p>Check JAVA_HOME</p> <p>Make sure your <code>JAVA_HOME</code> environment variable points to supported Java installation. Example:</p> <pre>\$ echo \$JAVA_HOME /opt/jdk1.8.0_60</pre>
---	---

2	<p>Change the management port for node PC1-1</p> <p>Edit the file <code>PC1-1/config/sgcenv</code> and change the <code>JMX_PORT</code> setting to <code>10111</code> .</p> <p>The <code>JMX_PORT</code> is the management port to which the command line management console will connect. It is not normally necessary to change this setting, but we must since we are running multiple nodes on a single system and they cannot both bind to the same port.</p> <div data-bbox="394 440 1885 656">  <p>Throughout this walk-through we will need a number of ports for different things. Any unique port numbers may be used, but it is helpful if there is some structure or pattern in place to help remember or calculate which port should be used in each situation. In this walk-through port numbers will be created as a concatenation of a purpose-specific prefix followed by the cluster number and the node number within that cluster. For management ports we're adopting the prefix <code>101</code> , therefore node <code>PC1-1</code> has management port <code>10111</code> , and node <code>PC2-1</code> has management port <code>10121</code> . If we were to add a second node to cluster <code>PC2</code> later on we would assign it <code>10122</code> to use as its <code>JMX_PORT</code> .</p> </div>
3	<p>Start node PC1-1</p> <pre data-bbox="394 748 1885 808">./PC1-1/bin/sgc start</pre> <p>If all is well, you should see:</p> <pre data-bbox="394 878 1885 938">SGC starting - daemonizing ... SGC started successfully</pre>
4	<p>Change the management port for node PC2-1</p> <p>Edit the file <code>PC2-1/config/sgcenv</code> and change the <code>JMX_PORT</code> setting to <code>10121</code> .</p> <p>The <code>JMX_PORT</code> is the management port to which the command line management console will connect. It is not normally necessary to change this setting, but we must since we are running multiple nodes on a single system and they cannot both bind to the same port.</p>
5	<p>Start node PC2-1</p> <pre data-bbox="394 1278 1885 1338">./PC2-1/bin/sgc start</pre> <p>If all is well, you should see:</p>


```
SGC starting - daemonizing ... SGC started successfully
```

If the SGC start command reported any errors please double check your `JAVA_HOME` environment variable and make sure that nothing has already bound the management ports 10111 and 10121 . If these ports are already in use on your system you may simply change them to something else and make a note of the values for later use.

3.6 Connect the management console

We now have two running OCSS7 clusters with blank configuration. The configuration we have done so far was done on a per-node basis using configuration files, but this does no more than give a node the minimal configuration it needs to boot and become a cluster member. The rest of our SGC configuration will now be done using the [Command-Line Management Console](#) on page 173 . Configuration done in this manner becomes cluster-wide configuration which is automatically propagated to and saved by every other cluster node, although for our single-node clusters that detail will not be particularly relevant.

It is recommended that you start one management console per node for this walk-through, however, if your system is low on RAM you may wish to start and stop these consoles as required.

1

Start the mangement console for PC1-1

```
./PC1-1/cli/sgc-cli.sh
```

Example:

```
$ ./PC1-1/cli/sgc-cli.sh Preparing to start SGC CLI ... Checking environment variables
[JAVA_HOME]=[/opt/jdk1.8.0_60] [CLI_HOME]=[/home/ocss7/quick-start/PC1-1/cli] Environment
is OK! Determining SGC home and JMX configuration [SGC_HOME]=/home/ocss7/quick-start/PC1-1
[JMX_HOST]=127.0.0.1 [JMX_PORT]=10111 Done +-----Environment-----
-----+ CLI_HOME: /home/ocss7/quick-start/PC1-1/cli JAVA: /opt/jdk1.8.0_60
JAVA_OPTS: -Dlog4j.configuration=file:/home/ocss7/quick-start/PC1-1/cli/conf/log4j.xml -
Dsgc.home=/home/ocss7/quick-start/PC1-1/cli +-----
-----+ SGC[127.0.0.1:10111]>
```

Here we can see the management console's prompt, which identifies the node to which it is connected by host and port.



The host and port settings were determined automatically by the CLI, which is possible because it is currently part of an SGC installation and can read the SGC configuration files. The CLI can also be copied out to elsewhere and run from another location or another host. When the CLI is not part of an SGC installation it is necessary to provide host and port options to the CLI on the command line.

2**Start the management console for PC2-1**

```
./PC2-1/cli/sgc-cli.sh
```

Example:

```
$ ./PC2-1/cli/sgc-cli.sh Preparing to start SGC CLI ... Checking environment variables
[JAVA_HOME]=[/opt/jdk1.8.0_60/] [CLI_HOME]=[/home/ocss7/quick-start/PC2-1/cli] Environment
is OK! Determining SGC home and JMX configuration [SGC_HOME]=/home/ocss7/quick-start/PC2-1
[JMX_HOST]=127.0.0.1 [JMX_PORT]=10121 Done +-----Environment-----
-----+ CLI_HOME: /home/ocss7/quick-start/PC2-1/cli JAVA: /opt/jdk1.8.0_6
0/ JAVA_OPTS: -Dlog4j.configuration=file:/home/ocss7/quick-start/PC2-1/cli/conf/log4j.xml
-Dsgc.home=/home/ocss7/quick-start/PC2-1/cli +-----
-----+ SGC[127.0.0.1:10121]>
```



The management console supports tab completion and suggestions. If you hit tab while in the console it will complete the command, parameter, or value as best it can. If the console is unable to complete the command, parameter, or value entirely because there are multiple completion choices then it will display the available choices.



You can exit the management console either by hitting `ctrl-d` or entering the `quit` command.

3.7 General configuration

[General Configuration](#) on page 93 is that which is fundamental to the cluster and the nodes within it. For our purposes this means:

- setting the local Point Codes for the *clusters* ,

- setting the basic communication attributes of each *node* .

The basic communication attributes of each node are used to control:

- payload message transfer between SGCs within the cluster; and
- communication with client TCAP stacks running in Rhino or the Scenario Simulator.



The distinction between *clusters* and *nodes* is about to become apparent because each *cluster* has exactly one local Point Code for which it provides services and which is set once for the entire cluster. In contrast, each *node* must be defined and given its own basic communication configuration.

1a	<p>Set the Point Code for PC1-1's cluster to 1</p> <p>Within the management console for PC1-1 run:</p> <pre>modify-parameters: sp=1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> modify-parameters: sp=1 OK parameters updated.</pre>
1b	<p>Set the Point Code for PC2-1's cluster to 2</p> <p>Within the management console for PC2-1 run:</p> <pre>modify-parameters: sp=2</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> modify-parameters: sp=2 OK parameters updated.</pre>
2a	<p>Configure node PC1-1's basic communication attributes</p> <p>Within the management console for PC1-1 run:</p>

```
create-node: oname=PC1-1, switch-local-address=127.0.0.1, switch-port=11011, stack-data-port=12011, stack-http-port=13011, enabled=true
```

Example:

```
SGC[127.0.0.1:10111]> create-node: oname=PC1-1, switch-local-address=127.0.0.1, switch-port=11011, stack-data-port=12011, stack-http-port=13011, enabled=true OK node created.
```



The value given to `oname` above must exactly match the value of `ss7.instance` which was set in [SGC cluster membership configuration](#) on page 22 . If the values are different the running node will assume this configuration is not intended for it, but for some other cluster node.

This command configures network communication for:

- message passing between SGC cluster members, through the attributes starting with `switch-` ; and
- communication with client TCAP stacks such as Rhino or the Scenario Simulator, through the attributes starting with `stack-` .



There are two attributes not specified in this command, `stack-data-address` and `stack-http-address` , which control the addresses used for client TCAP stack communications. These have been left to default to the value of `switch-local-address` because we are running everything on a single system. A typical production installation would partition the two traffic types by having one value for `switch-local-address` , and a different value shared between `stack-data-address` and `stack-http-address` . See [Network Architecture Planning](#) on page 63 and the [node](#) on page 93 configuration attributes for details.

2b

Configure node PC2-1's basic communication attributes

Within the management console for PC2-1 run:

```
create-node: oname=PC2-1, switch-local-address=127.0.0.1, switch-port=11021, stack-data-port=12021, stack-http-port=13021, enabled=true
```



This command differs from the previous command in that the ports have been carefully chosen not to conflict with those of the other SGC process. This is only necessary because all the SGC processes are running on a single test

system in this walk through. Typical production installations, where each host has only one SGC node running on it, could omit the ports entirely and use the default values on every node.

Example:

```
SGC[127.0.0.1:10121]> create-node: oname=PC2-1, switch-local-address=127.0.0.1, switch-port=11021, stack-data-port=12021, stack-http-port=13021, enabled=true OK node created.
```



The value given to `oname` above must exactly match the value of `ss7.instance` which was set in [SGC cluster membership configuration](#) on page 22 . If the values are different the running node will assume this configuration is not intended for it, but for some other cluster node.

Of the attributes we set above only the `switch-local-address` and `stack-data-port` settings are required for future configuration; we'll use them when we get to the [Scenario Simulator configuration](#) on page 47 section.



The `create-node` command is discussed above in the context of configuring the basic communication attributes to be used, but it also creates a node configuration object which can be enabled or disabled and for which the current state can be seen when using the `display-node` command. It has been discussed this way because some configuration *must* be provided, no matter what your configuration. If it was not necessary to provide some configuration then the SGC could simply automatically detect and add cluster nodes as they come online.

3.8 M3UA configuration

We will now begin configuring the M3UA layer of our network. There are a number of ways this can be done, but for the purposes of this walk-through we will use:

- a single Application Server (AS) between the two instances,
- the cluster for Point Code 1 as a client (in IPSP mode),
- the cluster for Point Code 2 as a server (in IPSP mode), and
- one SCTP association between the two nodes.

At a high level the procedure we're about to follow will:

- define the Application Server (AS) on each cluster,
- define routes to our destination Point Codes through the defined AS,
- define the SCTP connection on each node, and
- associate the SCTP connection with the Application Server.

All the steps below are in two parts, the part "a" commands must be run on the management console connected to node PC1-1 and the part "b" commands must be run on the management console connected to node PC2-1. If this becomes confusing please check the examples given, which will indicate the correct management console by the port number in the prompt.



Those familiar with M3UA will note that Single Exchange is used. The SGC does not support double exchange.

1a	<p>Define the Application Server for PC1-1</p> <p>Create the AS with <code>traffic-maintenance-role=ACTIVE</code> and Routing Context 2:</p> <pre>create-as: oname=PC2, traffic-maintenance-role=ACTIVE, rc=2, enabled=true</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-as: oname=PC2, traffic-maintenance-role=ACTIVE, rc=2, enabled=true OK as created.</pre>
1b	<p>Define the Application Server for PC2-1</p> <p>On PC2-1 note that <code>traffic-maintenance-role=PASSIVE</code>:</p> <pre>create-as: oname=PC2, traffic-maintenance-role=PASSIVE, rc=2, enabled=true</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-as: oname=PC2, traffic-maintenance-role=PASSIVE, rc=2, enabled=true OK as created.</pre>

2a**Define the local SCTP association's endpoint for PC1-1**

```
create-local-endpoint: oname=PC1-1-PC2-1, node=PC1-1, port=21121
```

Example:

```
SGC[127.0.0.1:10111]> create-local-endpoint: oname=PC1-1-PC2-1, node=PC1-1, port=21121 OK
local-endpoint created.
```

This defines a local endpoint which will be bound to SCTP port 21121 .

**Naming and oname**

Every configuration object has an object name field called `oname` . These onames serve both as user documentation and the method of referring to other configuration objects, as seen here, where the `node` attribute refers to our node's `oname` .


It is a good idea to plan a consistent and informative naming scheme before starting. In this walk-through several strategies are used:

- the owning cluster name is used where a cluster as a whole owns an object (the AS is named `PC2` because the cluster for PC 2 is acting as the server, and therefore has the greatest claim to ownership);
- the owning node name is used where a specific node owns an object (each node is named for itself, for example);
- where an object connects two things (like an SCTP association end point which will be used for outgoing connections) the name is the client node followed by the server node (for example `oname=PC1-1-PC2-1` above)

2b**Define the local SCTP association's endpoint for PC2-1**

```
create-local-endpoint: oname=PC2-1, node=PC2-1, port=22100
```

This defines a local endpoint which will be bound to SCTP port 22100 .

	<p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-local-endpoint: oname=PC2-1, node=PC2-1, port=22100 OK local-endpoint created.</pre> <div>  <p>The <code>oname</code> here is the same as the node's <code>oname</code> because the node owns this configuration object. In this walk-through we will only be using it to connect to PC1-1, but because we intend to have this node acting in the <i>server</i> role for SCTP we could reasonably expect more nodes from our peer cluster to connect to it, so it would be misleading to name it PC2-1-PC1-1 in the style used for step 2a.</p> </div>
3a	<p>Define the local SCTP endpoint IP addresses for PC1-1</p> <p>We will now define the IP address to be used by our SCTP association.</p> <pre>create-local-endpoint-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, local-endpoint-name=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-local-endpoint-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, local-endpoint-name=PC1-1-PC2-1 OK local-endpoint-ip created.</pre> <p>As you can see above, a <code>local-endpoint-ip</code> associates itself with <i>one</i> particular <code>local-endpoint</code> by setting <code>local-endpoint-name</code> to the <code>oname</code> value of the intended <code>local-endpoint</code>. This step is necessary because SCTP supports "multi-homing", meaning that one association can be bound to multiple local IP addresses. Typically these IP addresses would be associated with resilient physical network paths, allowing multi-homing to provide protection against network failure.</p>
3b	<p>Define the local SCTP endpoint IP addresses for PC2-1</p> <p>Similar to 3a, above:</p> <pre>create-local-endpoint-ip: oname=PC2-1, ip=127.0.0.1, local-endpoint-name=PC2-1</pre> <p>Example:</p>

	<pre>SGC[127.0.0.1:10121]> create-local-endpoint-ip: oname=PC2-1, ip=127.0.0.1, local-endpoint-name=PC2-1 OK local-endpoint-ip created.</pre>
4a	<p>Enable the local endpoint for PC1-1</p> <p>The local endpoint was created in its default <code>enabled=false</code> to allow us to add local endpoint IP addresses to it. The SGC does not allow changes to enabled local endpoints to avoid unexpected service interruptions while it tears down the connection and establishes it with new configuration. We are now done modifying this configuration, so it is time to enable the local endpoint:</p> <pre>enable-local-endpoint: oname=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> enable-local-endpoint: oname=PC1-1-PC2-1 OK local-endpoint enabled.</pre>
4b	<p>Enable the local endpoint for PC2-1</p> <pre>enable-local-endpoint: oname=PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> enable-local-endpoint: oname=PC2-1 OK local-endpoint enabled.</pre>
5a	<p>Define the client connection for PC1-1 to PC2-1</p> <p>We will now define the SCTP association used by PC1-1, as well as some M3UA settings for the connection:</p> <pre>create-connection: oname=PC1-1-PC2-1, port=22100, local-endpoint-name=PC1-1-PC2-1, conn-type=CLIENT, state-maintenance-role=ACTIVE, is-ipsp=true, enabled=true</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-connection: oname=PC1-1-PC2-1, port=22100, local-endpoint-name=PC1-1-PC2-1, conn-type=CLIENT, state-maintenance-role=ACTIVE, is-ipsp=true, enabled=true OK connection created.</pre>

	<p>The port here is the <i>remote</i> SCTP port to which the SGC should connect, the local IP and port information comes from the <code>local-endpoint-name</code>. This connection will act in all ways as a "client" connection, in that it will initiate the connection and begin the conversation. If you're interested in the exact details please see the connection reference documentation on page 104.</p>
5b	<p>Define the server connection for PC2-1 from PC1-1</p> <p>Similar to the above, this defines a client connection to the node, which is acting as a server:</p> <pre>create-connection: oname=PC1-1-PC2-1, port=21121, local-endpoint-name=PC2-1, conn-type=SERVER, state-maintenance-role=PASSIVE, is-ipsp=true, enabled=true</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-connection: oname=PC1-1-PC2-1, port=21121, local-endpoint-name=PC2-1, conn-type=SERVER, state-maintenance-role=PASSIVE, is-ipsp=true, enabled=true OK connection created.</pre> <p>the port in this command is the remote port from which the connection will be initiated. It must match the configuration in node PC1-1 or the connection will not be accepted.</p>
6a	<p>Define the connection IP addresses for PC1-1 to PC2-1</p> <p>Just as we had to define local endpoint IP addresses earlier, we must now define the remote connection IP addresses to which the node should connect:</p> <pre>create-conn-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, conn-name=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-conn-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, conn-name=PC1-1-PC2-1 OK conn-ip created.</pre> <p>Again, this extra step is because SCTP supports multi-homing on page .</p>
6b	<p>Define the connection IP addresses for PC2-1 from PC1-1</p>

	<p>The compliment of step 6a, above, PC2-1 needs to know which IP addresses to expect a connection from:</p> <pre>create-conn-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, conn-name=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-conn-ip: oname=PC1-1-PC2-1, ip=127.0.0.1, conn-name=PC1-1-PC2-1 OK conn-ip created.</pre> <p>The IP address here must match the <code>local-endpoint-ip</code> address from PC1-1 or the connection will not be accepted by PC2-1.</p>
7a	<p>Connect the AS to the connection on PC1-1</p> <p>We must now tell the SGC that our AS should use the connection we have defined:</p> <pre>create-as-connection: oname=PC1-1-PC2-1, as-name=PC2, conn-name=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-as-connection: oname=PC1-1-PC2-1, as-name=PC2, conn-name=PC1-1-PC2-1 OK as-connection created.</pre> <p>This <code>as-connection</code> is necessary because one AS may use many connections, and a connection may serve many Application Servers, in an "many-to-many" relationship.</p>
7b	<p>Connect the AS to the connection on PC2-1</p> <pre>create-as-connection: oname=PC1-1-PC2-1, as-name=PC2, conn-name=PC1-1-PC2-1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-as-connection: oname=PC1-1-PC2-1, as-name=PC2, conn-name=PC1-1-PC2-1 OK as-connection created.</pre>
8a	<p>Define the route on PC1-1 to Point Code 2</p>

	<p>The final step, we must now define which Destination Point Codes can be reached via our Application Server. Define a Destination Point Code for PC=2 and a route to it via our AS with the following commands:</p> <pre>create-dpc: oname=PC2, dpc=2 create-route: oname=PC2, as-name=PC2, dpc-name=PC2</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-dpc: oname=PC2, dpc=2 OK dpc created. SGC[127.0.0.1:10111]> create-route: oname=PC2, as-name=PC2, dpc-name=PC2 OK route created.</pre>
8b	<p>Define the route on PC2-1 to Point Code 1</p> <p>Define a Destination Point Code for PC=1 and a route to it via our AS with the following commands:</p> <pre>create-dpc: oname=PC1, dpc=1 create-route: oname=PC1, as-name=PC2, dpc-name=PC1</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-dpc: oname=PC1, dpc=1 OK dpc created. SGC[127.0.0.1:10121]> create-route: oname=PC1, as-name=PC2, dpc-name=PC1 OK route created.</pre>

General and M3UA configuration is now complete. In the next section we will check that everything is working correctly.

3.9 M3UA state inspection

You should now have two SGCs which are connected to each other at the M3UA layer. Before we move on to the upper layers of configuration we should check that everything is working as expected up to this point. If you are confident of your setup and in a hurry you can skip this section.

Please note that it is not normally necessary to check state in this exhaustive a manner, we are doing it in this step-by-step fashion to provide some familiarization with the SGC state inspection facilities and assist with troubleshooting.



Most of the commands shown below show both the definition and the state of the various configuration objects they examine, and are intended for those modifying or considering modifying the configuration of the SGC. If you are interested strictly in state rather than

configuration, there is a related family of commands which start with `display-info-` which will show extended state information without any configuration details.

1	<p>Check the <code>display-active-alarms</code> command for problems</p> <p>The <code>display-active-alarms</code> command can show problems from any aspect of the SGC's operation. If you check it now you should see:</p> <p>PC1-1</p> <pre>SGC[127.0.0.1:10111]> display-active-alarm: Found 0 objects.</pre> <p>PC2-1</p> <pre>SGC[127.0.0.1:10121]> display-active-alarm: Found 0 objects.</pre> <p>If, instead, you see one or more alarms, don't worry, we'll step through the diagnostics one by one.</p>
2	<p>Check the node state</p> <p>If something is wrong with the node state or configuration than nothing will work. Run</p> <pre>display-node</pre> <p>on both nodes. Both nodes should say that the <code>active</code> state is <code>true</code> . If your <code>active</code> state is not <code>true</code> then either:</p> <ul style="list-style-type: none"> the <code>enabled</code> attribute is set to <code>false</code> , and you need to use the <code>enable-node</code> command to enable it; or the <code>ss7.instance</code> on page 22 values does not match the node's <code>oname</code> on page 26 value.
3	<p>Check the local endpoint state</p> <p>The local endpoint must be enabled and active before the connection between the nodes will work. Run:</p> <pre>display-local-endpoint</pre> <p>on both nodes. Both nodes should say that the <code>active</code> state is <code>true</code> . If the <code>active</code> state is not <code>true</code> then:</p>

	<ul style="list-style-type: none"> the enabled attribute is set to <code>false</code> , which can be corrected with the <code>enable-local-endpoint</code> command.
4	<p>Check the connection state</p> <p>The next thing to check, working up the stack, is the SCTP association. Run</p> <pre>display-connection</pre> <p>on both nodes. Both nodes should say that the <code>active</code> state is <code>true</code> . If the <code>active</code> state is not <code>true</code> then either:</p> <ul style="list-style-type: none"> the enabled attribute is set to <code>false</code> on one or the other of the nodes, and you need to use the <code>enable-connection</code> command to enable it; there is a configuration mismatch between the nodes with regard to ports or IP addresses; or the ports selected in this walk through may not work correctly due to your local operating system configuration, in which case you may need to consult Configuring network features on page 52 . <p>It is often helpful to consult either the active alarms list or the logs when diagnosing connection issues, but that is outside the scope of this walk-through.</p>
5	<p>Check the AS state</p> <p>The AS should be active on both nodes. Run:</p> <pre>display-as</pre> <p>on both nodes to check. The state should be listed as <code>ACTIVE</code> . If the state is not <code>ACTIVE</code> then:</p> <ul style="list-style-type: none"> there is a problem at a lower layer; the two clusters disagree about the <code>rc</code> value; or one of the <code>create-as-connection</code> commands was omitted.
6	<p>Check the SCCP state</p> <p>SCCP is the next layer up, and we have not yet configured it, but it should be able to activate and communicate with its peer at this point. Run this command on both nodes to check:</p>

```
display-info-remotessninfo
```

This should show the following output on both nodes:

```
SGC[127.0.0.1:10111]> display-info-remotessninfo Found 2 object(s): +-----+-----+
+-----+ |dpc |ssn |status | +-----+-----+-----+ |1 |1 |ALLOWED
| +-----+-----+-----+ |2 |1 |ALLOWED | +-----+-----+
-----+
```

This output shows that the SCCP layers on each node are communicating with each other.



SSN=1 is the SCCP management SubSystem Number. If the status of SSN=1 is not `ALLOWED` then the SCCP layers are unable to communicate with each other and no other SSN will be reachable.

If the status shown above is `PROHIBITED` for the remote Destination Point Code then:

- the [route entries](#) on page [112](#) are probably incorrect or missing.

3.10 SCCP configuration

In [The Plan](#) on page 20 we can see that the two Scenario Simulators expect to refer to each other by their global titles as follows:

- 1234: PC=1,SSN=101
- 4321: PC=2,SSN=102

Several inbound and outbound global title translation (GTT) rules are required to allow this to happen, which we will create now.

Also, while not technically necessary, we will configure [Concerned Point Codes](#) on page 112 for each of the two nodes, so that they will inform each other about changes to the state of interesting SSNs.

All the steps below are in two parts, the part "a" commands must be run on the management console connected to node PC1-1 and the part "b" commands must be run on the management console connected to node PC2-1. If this becomes confusing please check the examples given, which will indicate the correct management console by the port number in the prompt.

1a	<p>Outbound GTT setup on PC1-1</p> <p>Run the following commands to setup outbound global title translation on PC1-1:</p> <pre>create-outbound-gt: oname=4321, addrinfo=4321 create-outbound-gtt: oname=4321, gt=4321, dpc=2, priority=5</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10111]> create-outbound-gt: oname=4321, addrinfo=4321 OK outbound-gt created. SGC[127.0.0.1:10111]> create-outbound-gtt: oname=4321, gt=4321, dpc=2, priority=5 OK outbound-gtt created.</pre> <p>This defines a Global Title and then creates a translation rule which will cause messages with that GT in the Called Party Address to be routed to our peer at PC=2.</p>
1b	<p>Outbound GTT setup on PC2-1</p> <p>Run the following commands to setup outbound global title translation on PC2-1:</p> <pre>create-outbound-gt: oname=1234, addrinfo=1234 create-outbound-gtt: oname=1234, gt=1234, dpc=1, priority=5</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-outbound-gt: oname=1234, addrinfo=1234 OK outbound-gt created. SGC[127.0.0.1:10121]> create-outbound-gtt: oname=1234, gt=1234, dpc=1, priority=5 OK outbound-gtt created.</pre> <p>This defines a Global Title and then creates a translation rule which will cause messages with that GT in the Called Party Address to be routed to our peer at PC=1.</p>
2a	<p>Inbound GTT setup on PC1-1</p> <p>Run the following to setup inbound GTT on PC1-1:</p>

	<pre>create-inbound-gtt: oname=1234, addrinfo=1234, ssn=101 create-outbound-gt: oname=1234, addrinfo=1234 create-outbound-gtt: oname=1234, gt=1234, dpc=1, priority=5</pre> <p>Example</p> <pre>SGC[127.0.0.1:10111]> create-inbound-gtt: oname=1234, addrinfo=1234, ssn=101 OK inbound-gtt created. SGC[127.0.0.1:10111]> create-outbound-gt: oname=1234, addrinfo=1234 OK outbound-gt created. SGC[127.0.0.1:10111]> create-outbound-gtt: oname=1234, gt=1234, dpc=1, priority=5 OK outbound-gtt created.</pre> <p>The first command creates an inbound GTT rule for the Global Title we expect to be accepted traffic on. The second and third commands may look somewhat surprising, as they create an outbound GTT rule. This is the correct configuration for our network, as SCCP's service messages (UDTS and XUDTS) may be generated locally in response to traffic we are attempting to send, and these service messages are routed as outbound messages.</p>
2b	<p>Inbound GTT setup on PC2-1</p> <p>Run the following to setup inbound GTT on PC2-1:</p> <pre>create-inbound-gtt: oname=4321, addrinfo=4321, ssn=102 create-outbound-gt: oname=4321, addrinfo=4321 create-outbound-gtt: oname=4321, gt=4321, dpc=2, priority=5</pre> <p>Example:</p> <pre>GC[127.0.0.1:10121]> create-inbound-gtt: oname=4321, addrinfo=4321, ssn=102 OK inbound-gtt created. SGC[127.0.0.1:10121]> create-outbound-gt: oname=4321, addrinfo=4321 OK outbound-gt created. SGC[127.0.0.1:10121]> create-outbound-gtt: oname=4321, gt=4321, dpc=2, priority=5 OK outbound-gtt created.</pre>
3a	<p>Create the Concerned Point Code on PC1-1</p> <p>Run the following to configure PC1-1 to announce SSN changes for SSN=101 to the PC2 cluster:</p> <pre>create-cpc: oname=PC2-101, dpc=2, ssn=101</pre> <p>Example:</p>

	SGC[127.0.0.1:10111]> create-cpc: oname=PC2-101, dpc=2, ssn=101 OK cpc created.
3b	<p>Create the Concerned Point Code on PC2-1</p> <p>Run the following to configure PC2-1 to announce SSN changes for SSN=102 to the PC1 cluster:</p> <pre>create-cpc: oname=PC1-102, dpc=1, ssn=102</pre> <p>Example:</p> <pre>SGC[127.0.0.1:10121]> create-cpc: oname=PC1-102, dpc=1, ssn=102 OK cpc created.</pre>

This completes our SCCP configuration, which we will check in the next section.

3.11 SCCP state inspection

We now have two fully configured SCCP layers. We will now check their state to make sure they will work as expected.

1	<p>Check the outbound GTT state on PC1-1</p> <p>The following command will show the current state of configured outbound GTT rules:</p> <pre>display-info-ogtinfo: column=addrInfo, column=connId, column=rc, column=dpc</pre> <p>Example on PC1-1</p> <pre>SGC[127.0.0.1:10111]> display-info-ogtinfo: column=addrInfo, column=connId, column=rc, column=dpc Found 2 object(s): +-----+-----+-----+-----+ addrInfo connId rc dpc +-----+-----+-----+-----+ 1234 -1 1 +-----+-----+-----+-----+ 4321 PC1-1-PC2-1 2 2 +-----+-----+-----+-----+</pre> <p>For GT 1234 we can see that:</p> <ul style="list-style-type: none"> it has no associated connection, and no valid routing context (-1), and
----------	--

- the DPC to be used is 1 , which is our local PC.

This GT will be routed to the local SGC.

For GT 4321 we can see that:

- it has an associated connection and Routing Context, and
- the DPC to be used is 2 , which is the remote PC.

This GT will be routed to PC2-1 using the specified connection and Routing Context.

Example on PC2-1

```
SGC[127.0.0.1:10121]> display-info-ogtinfo: column=addrInfo, column=connId, column=rc,
column=dpc Found 2 object(s): +-----+-----+-----+-----+ |
addrInfo |connId |rc |dpc | +-----+-----+-----+-----+ |1234 |
PC1-1-PC2-1 |2 |1 | +-----+-----+-----+-----+ |4321 | |-1 |2 |
+-----+-----+-----+-----+
```



All `display-` commands can be given a list of `column=` arguments to restrict the columns listed. This feature has been used in the examples above to reduce clutter and make the output more readable.

2

Check the local SSN state

The command:

```
display-info-localssninfo
```

will list the state of all SSNs which are either:

- declared in Concerned Point Code configuration, or
- known from current or previous SSN connections.

Example on PC1-1

```
GC[127.0.0.1:10111]> display-info-localssninfo: column=ssn, column=status Found 2 object(s):
+-----+-----+ |ssn |status | +-----+-----+ |1 |ALLOWED |
+-----+-----+ |101 |PROHIBITED | +-----+-----+
```

Example on PC2-1

```
SGC[127.0.0.1:10121]> display-info-localssninfo: column=ssn, column=status Found 2 object(s):
+-----+-----+ |ssn |status | +-----+-----+ |1 |ALLOWED |
+-----+-----+ |102 |PROHIBITED | +-----+-----+
```

3.12 Scenario Simulator installation

This quick start walk-through will use the OC Scenario Simulator to test the network, rather than Rhino with CGIN, for simplicity.

For this quick start we will be assuming that your Scenario Simulator package is shipped with an IN Scenario Pack which does not support OCSS7 (which is true for Scenario Simulator 2.2.0.x), or with an obsolete version of the IN Scenario Pack. If you know that your Scenario Simulator contains a suitable IN Scenario Pack you may skip this section after completing it through step 2.

1**Unpack the Scenario Simulator archive file**

```
unzip scenario-simulator-package-VERSION.zip
```

(replacing `scenario-simulator-package-VERSION.zip` with the correct file name).

This creates the distribution directory, `scenario-simulator-VERSION`, in the current working directory.

Example:

```
$ unzip scenario-simulator-package-2.3.0.6.zip Archive: scenario-simulator-package-2.3.0.6
.zip creating: scenario-simulator-2.3.0.6/ creating: scenario-simulator-2.3.0.6/licenses/
inflating: scenario-simulator-2.3.0.6/licenses/LICENSE-XPathOverSchema.txt inflating:
scenario-simulator-2.3.0.6/licenses/LICENSE-antlr.txt [...]
```

2**Change directory into the Scenario Simulator directory**

```
cd scenario-simulator-VERSION
```

(replacing `scenario-simulator-package-VERSION.zip` with the correct file name).

Example:

```
$ cd scenario-simulator-2.3.0.6/
```

3

Install the new IN Scenario Pack

We want to replace the old IN Scenario Pack with the new, which can be done with the following commands. Please ensure that you are in the Scenario Simulator's installation directory before running these commands.

```
rm -r in-examples/ protocols/in-scenario-pack-* unzip -o ../in-scenario-pack-VERSION.zip
```

(replacing `../in-scenario-pack-VERSION.zip` with the correct file name). The `-o` option is being used here to automatically overwrite existing files without prompting, which is desirable in this case since we expect to replace certain files which were not explicitly removed with the `rm` command above.

Example:

```
$ rm -r in-examples/ protocols/in-scenario-pack-* $ unzip -o ../in-scenario-pack-1.5.3.1.zip
Archive: ../in-scenario-pack-1.5.3.1.zip inflating: protocols/in-scenario-pack-1.5.3.jar
creating: in-examples/ creating: in-examples/2sims/ creating: in-examples/2sims/config/
creating: in-examples/2sims/config/loopback/ creating: in-examples/2sims/config/mach7/
creating: in-examples/2sims/config/ocss7/ creating: in-examples/2sims/config/signalware/
creating: in-examples/2sims/scenarios/ creating: in-examples/3sims/ creating: in-examples/3sims/config/
creating: in-examples/3sims/config/loopback/ creating: in-examples/3sims/config/mach7/
creating: in-examples/3sims/config/ocss7/ creating: in-examples/3sims/config/signalware/
creating: in-examples/3sims/scenarios/ inflating: CHANGELOGS/CHANGELOG-in.txt
inflating: README/README-in.txt inflating: in-examples/2sims/config/loopback/cgin-tcap
sim-endpoint1.properties inflating: in-examples/2sims/config/loopback/cgin-tcapsim-endpoint2.properties
inflating: in-examples/2sims/config/loopback/setup-sim1.commands inflating: in-examples/2sims/config/loopback/setup-sim2.commands
inflating: in-examples/2sims/config/loopback/tcapsim-gt-table.txt inflating: in-examples/2sims/config/mach7/mach7-endpoint1.properties
inflating: in-examples/2sims/config/mach7/mach7-endpoint2.properties inflating: in-examples/2sims/config/mach7/setup-mach7-endpoint1.commands
inflating: in-examples/2sims/config/mach7/setup-mach7-endpoint2.commands inflating: in-examples/2sims/config/ocss7/ocss7-endpoint1.properties
inflating: in-examples/2sims/config/ocss7/ocss7-endpoint2.properties
```

```

s inflating: in-examples/2sims/config/ocss7/setup-sim-endpoint1.commands inflating: in-
examples/2sims/config/ocss7/setup-sim-endpoint2.commands inflating: in-examples/2sims/
config/setup-examples-sim1.commands inflating: in-examples/2sims/config/setup-examples-
sim2.commands inflating: in-examples/2sims/config/signalware/setup-signalware-endpoint1.co
mmands inflating: in-examples/2sims/config/signalware/setup-signalware-endpoint2.commands
inflating: in-examples/2sims/config/signalware/signalware-endpoint1.properties inflating:
in-examples/2sims/config/signalware/signalware-endpoint2.properties inflating: in-examples
/2sims/scenarios/CAPv3-Demo-ContinueRequest.scen inflating: in-examples/2sims/scenarios/CA
Pv3-Demo-ReleaseCallRequest.scen inflating: in-examples/2sims/scenarios/INAP-SSP-SCP.scen
inflating: in-examples/3sims/config/loopback/cgin-tcapsim-endpoint1.properties inflating:
in-examples/3sims/config/loopback/cgin-tcapsim-endpoint2.properties inflating: in-examples
/3sims/config/loopback/cgin-tcapsim-endpoint3.properties inflating: in-examples/3sims/
config/loopback/setup-sim1.commands inflating: in-examples/3sims/config/loopback/setup-
sim2.commands inflating: in-examples/3sims/config/loopback/setup-sim3.commands inflating:
in-examples/3sims/config/loopback/tcapsim-gt-table.txt inflating: in-examples/3sims/con
fig/mach7/mach7-endpoint1.properties inflating: in-examples/3sims/config/mach7/mach7-
endpoint2.properties inflating: in-examples/3sims/config/mach7/mach7-endpoint3.properties
inflating: in-examples/3sims/config/mach7/setup-mach7-endpoint1.commands inflating: in-
examples/3sims/config/mach7/setup-mach7-endpoint2.commands inflating: in-examples/3sims/
config/mach7/setup-mach7-endpoint3.commands inflating: in-examples/3sims/config/ocss7/ocss
7-endpoint1.properties inflating: in-examples/3sims/config/ocss7/ocss7-endpoint2.propertie
s inflating: in-examples/3sims/config/ocss7/ocss7-endpoint3.properties inflating: in-
examples/3sims/config/ocss7/setup-sim-endpoint1.commands inflating: in-examples/3sims/
config/ocss7/setup-sim-endpoint2.commands inflating: in-examples/3sims/config/ocss7/setup-
sim-endpoint3.commands inflating: in-examples/3sims/config/setup-examples-sim1.commands
inflating: in-examples/3sims/config/setup-examples-sim2.commands inflating: in-examples/3si
ms/config/setup-examples-sim3.commands inflating: in-examples/3sims/config/signalware/setu
p-signalware-endpoint1.commands inflating: in-examples/3sims/config/signalware/setup-signa
lware-endpoint2.commands inflating: in-examples/3sims/config/signalware/setup-signalware-e
ndpoint3.commands inflating: in-examples/3sims/config/signalware/signalware-endpoint1.prop
erties inflating: in-examples/3sims/config/signalware/signalware-endpoint2.properties
inflating: in-examples/3sims/config/signalware/signalware-endpoint3.properties inflating:
in-examples/3sims/scenarios/CAPv2-Relay.scen inflating: in-examples/3sims/scenarios/
INAP-SSP-SCP-HLR.scen inflating: in-examples/3sims/scenarios/MAP-MT-SMS-DeliveryAbsentSubs
criber.scen inflating: in-examples/3sims/scenarios/MAP-MT-SMS-DeliveryPresentSubscriber.sc
en inflating: in-examples/README-in-examples.txt inflating: licenses/LICENSE-netty.txt
inflating: licenses/LICENSE-slf4j.txt inflating: licenses/README-LICENSES-in-scenario-
pack.txt

```

3.13 Scenario Simulator configuration

We will now configure two Scenario Simulator instances and connect them to the cluster. This work should be done in the Scenario Simulator installation directory, which is where the steps from the previous section left us.



The Scenario Simulator and CGIN use identical configuration properties and values when using OCSS7, the only difference between the two is the procedure used for setup and configuration.

1	<p>Set the OCSS7 connection properties for Simulator 1</p> <p>Edit the file <code>in-examples/2sims/config/ocss7/ocss7-endpoint1.properties</code> and make the following changes:</p> <pre>local-sccp-address = type=C7,ri=gt,ssn=101,digits=1234,national=true</pre> <p>and</p> <pre>ocss7.sgcs = 127.0.0.1:12011</pre> <p>The port in the <code>ocss7.sgcs</code> property is the <code>stack-data-port</code> we configured earlier on page for node PC1-1.</p>
2	<p>Set the OCSS7 connection properties for Simulator 2</p> <p>Edit the file <code>in-examples/2sims/config/ocss7/ocss7-endpoint2.properties</code> and make the following changes:</p> <pre>local-sccp-address = type=C7,ri=gt,ssn=102,digits=4321,national=true</pre> <p>and</p> <pre>ocss7.sgcs = 127.0.0.1:12021</pre> <p>The port in the <code>ocss7.sgcs</code> property is the <code>stack-data-port</code> we configured earlier on page for node PC2-1.</p>
3	<p>Set the Scenario Simulator endpoint addresses</p> <p>Edit the following files:</p>

- `in-examples/2sims/config/ocss7/setup-sim-endpoint1.commands`
- `in-examples/2sims/config/ocss7/setup-sim-endpoint2.commands`

and replace the two lines beginning:

```
set-endpoint-address endpoint1 set-endpoint-address endpoint2
```

with

```
set-endpoint-address endpoint1 type=c7,ri=gt,pc=1,ssn=101,digits=1234,national=true set-  
endpoint-address endpoint2 type=c7,ri=gt,pc=2,ssn=102,digits=4321,national=true
```



We've given the PC and SSN information in the above endpoint addresses, and you may be wondering why and whether global title translation is actually happening. The reason for this is that the Scenario Simulator is just that, a simulator, and it needs the PC and SSN information to work out roles and endpoints correctly for scenario validation. If you wish, you can change the SSN given to the simulator in `set-endpoint-address` to some other value without changing the value used by the TCAP stack in `local-sccp-address` and our test scenario will still work correctly because global title translation is actually happening and the inbound SSN is ignored by the receiving SGC.

The Scenario Simulators are now fully configured and ready to test our network.

3.14 Test the network

We will now test the network using the OpenCloud Scenario Simulator and one of the example IN scenarios included with it.

1

Start the scenario simulators

We need two Scenario Simulator instances for this test, one to initiate our test traffic, and one to respond. Start them with these two commands:

```
./scenario-simulator.sh -f in-examples/2sims/config/ocss7/setup-sim-endpoint1.commands -f in-  
examples/2sims/config/setup-examples-sim1.commands
```


and

```
./scenario-simulator.sh -f in-examples/2sims/config/ocss7/setup-sim-endpoint2.commands -f in-examples/2sims/config/setup-examples-sim2.commands
```

Example for Simulator 1:

```
$ ./scenario-simulator.sh -f in-examples/2sims/config/ocss7/setup-sim-endpoint1.commands -f in-examples/2sims/config/setup-examples-sim1.commands Starting JVM... Processing commands from file at in-examples/2sims/config/ocss7/setup-sim-endpoint1.commands Processing command: set-endpoint-address endpoint1 type=C7,ri=gt,digits=1234 Processing command: set-endpoint-address endpoint2 type=C7,ri=gt,digits=4321 Processing command: create-local-endpoint endpoint1 cgin -propsfile in-examples/2sims/config/ocss7/ocss7-endpoint1.properties Initializing local endpoint "endpoint1" ... Local endpoint initialized. Finished reading commands from file Processing commands from file at in-examples/2sims/config/setup-examples-sim1.commands Processing command: bind-role SSP-Loadgen endpoint1 Processing command: bind-role SCP-Rhino endpoint2 Processing command: wait-until-operational 60000 Simulator is operational Processing command: load-scenario in-examples/2sims/scenarios/INAP-SSP-SCP.scen Playing role "SSP-Loadgen" in initiating scenario "INAP-SSP-SCP" with dialogs [SSP-SCP] Processing command: load-scenario in-examples/2sims/scenarios/CAPv3-Demo-ContinueRequest.scen Playing role "SSP-Loadgen" in initiating scenario "CAPv3-Demo-ContinueRequest" with dialogs [SSP-SCP] Processing command: load-scenario in-examples/2sims/scenarios/CAPv3-Demo-ReleaseCallRequest.scen Playing role "SSP-Loadgen" in initiating scenario "CAPv3-Demo-ReleaseCallRequest" with dialogs [SSP-SCP] Finished reading commands from file Ready to start Please type commands... (type "help" <ENTER> for command help) >
```

Example for Simulator 2:

```
$ ./scenario-simulator.sh -f in-examples/2sims/config/ocss7/setup-sim-endpoint2.commands -f in-examples/2sims/config/setup-examples-sim2.commands Starting JVM... Processing commands from file at in-examples/2sims/config/ocss7/setup-sim-endpoint2.commands Processing command: set-endpoint-address endpoint1 type=C7,ri=gt,digits=1234 Processing command: set-endpoint-address endpoint2 type=C7,ri=gt,digits=4321 Processing command: create-local-endpoint endpoint2 cgin -propsfile in-examples/2sims/config/ocss7/ocss7-endpoint2.properties Initializing local endpoint "endpoint2" ... Local endpoint initialized. Finished reading commands from file Processing commands from file at in-examples/2sims/config/setup-examples-sim2.commands Processing command: bind-role SSP-Loadgen endpoint1 Processing command: bind-role SCP-Rhino endpoint2 Processing command: wait-until-operational 60000 Simulator
```

	<pre>is operational Processing command: load-scenario in-examples/2sims/scenarios/INAP-SSP-SCP .scen Playing role "SCP-Rhino" in receiving scenario "INAP-SSP-SCP" with dialogs [SSP-SCP] Processing command: load-scenario in-examples/2sims/scenarios/CAPv3-Demo-ContinueRequest. .scen Playing role "SCP-Rhino" in receiving scenario "CAPv3-Demo-ContinueRequest" with dialogs [SSP-SCP] Processing command: load-scenario in-examples/2sims/scenarios/CAPv3-Demo-Rel easeCallRequest.scen Playing role "SCP-Rhino" in receiving scenario "CAPv3-Demo-ReleaseCal lRequest" with dialogs [SSP-SCP] Finished reading commands from file Ready to start Please type commands... (type "help" <ENTER> for command help) ></pre>
2	<p>Check the remote SSN information</p> <p>Before running a test session let's pause to check the</p> <pre>display-info-remotessninfo</pre> <p>command, which should now show the following on both nodes:</p> <pre>SGC[127.0.0.1:10111]> display-info-remotessninfo Found 4 object(s): +-----+-----+ +-----+ dpc ssn status +-----+-----+-----+ 1 1 ALLOWED +-----+-----+-----+ 1 101 ALLOWED +-----+-----+ -----+ 2 1 ALLOWED +-----+-----+-----+ 2 102 ALLOWED +-----+-----+-----+</pre> <p>From this we can see that both SGCs have registered the connected simulators and informed the Concerned Point Codes about the state change for the SSN used by the simulator.</p>
3	<p>Run a test session</p> <p>On Simulator 1 run:</p> <pre>run-session CAPv3-Demo-ContinueRequest</pre> <p>This will run the test scenario, which is a basic CAPv3 IDP / CON scenario.</p> <p>Example:</p> <pre>> run-session CAPv3-Demo-ContinueRequest Send --> OpenRequest to endpoint2 Send --> InitialDP (Request) to endpoint2 Send --> Delimiter to endpoint2 Recv <-- OpenAccept from endpoint2</pre>

Recv <-- Continue (Request) from endpoint2 Recv <-- Close from endpoint2 Outcome of "CAPv3-Demo-ContinueRequest" session: Matched scenario definition "CAPv3-Demo-ContinueRequest"



This scenario has a 10 second delay between the `OpenRequest` and the `OpenAccept` so do not be concerned if it seems to be taking a little while to get a reply.

4 Installing the SGC

This section of the manual covers the installation, basic configuration, and control the SGC component of OCSS7.



The Rhino-side component of OCSS7, the TCAP stack, is automatically installed with CGIN and has no special installation procedure or requirements beyond those of Rhino and CGIN. Information on configuring the TCAP stack component can be found in [TCAP Stack configuration](#) on page 199 .

4.1 Checking prerequisites

Before installing OCSS7, make sure you have the required:

- [hardware](#) ,
- [operating system](#) , with `lksctp-tools` installed, and
- [Java Virtual Machine](#) .

Before attempting a production installation please also see [Network Architecture Planning](#) on page 63 .

4.2 Configuring network features



See also [Network Architecture Planning](#) on page 63

Before installing OCSS7, please configure the following network features:

Feature	What to configure
IP address	Make sure the system has an IPv4 or IPv6 address and is visible on the network.
Host names	Make sure that the system can resolve the localhost to loopback interface.

Multicast addresses (firewall rules)	<p>If the local system has a firewall installed, modify its rules to allow multicast UDP traffic:</p> <ul style="list-style-type: none"> • Multicast addresses are, by definition, in the range <code>224.0.0.0/4</code> (<code>224.0.0.0 - 239.255.255.255</code>) or <code>ff00::/8</code> for IPv6. OCSS7 by default uses address <code>224.2.2.3</code> . • OCSS7 uses multicast UDP to discover cluster members. The default port number used is <code>14327</code> . • Ensure that the firewall is configured to allow multicast messages through on the multicast address/port used by OCSS7.
Unicast addresses (firewall rules)	<p>If the local system has a firewall installed, modify its rules to allow TCP traffic:</p> <ul style="list-style-type: none"> • By default, OCSS7 listens on all available network interfaces for incoming TCP connections from other cluster members (this connection is used to distribute cluster configuration and run-time information state). By default, the port used is chosen dynamically, starting at port <code>5701</code> and incremented by 1 until a free port is found. • Addresses and ports to be used for message traffic distribution between nodes and the SGC-Stack-to-TCAP-Stack communication are user-configurable. They are configured when defining a new node in the SGC Cluster, as described in General Configuration on page 93

OCSS7 network configuration related to cluster operation can be customized by providing a `hazelcast.xml` configuration file in the `config` subdirectory of the OCSS7 distribution (for example, rename `config/hazelcast.xml.sample` to `config/hazelcast.xml`). For a description of possible configuration options, and the format of the `hazelcast.xml` configuration file, please see the [Hazelcast In-Memory Data Grid Documentation](#) (Version 2.6).



IPv6 considerations When using IPv6 addressing, remember to configure the `PREFER_IPV4` property in the `SGC_HOME/config/sgcenv` file. For details, please see [Configuring SGC_HOME/config/sgcenv](#) on page 59 .

4.3 User process tuning

Ensure that the user that OCSS7 will be running under has a soft limit of no less than 4096 user processes.



The number of permitted user processes may be determined at runtime using the `ulimit` command; for example `ulimit -Su`

This value may be changed by editing `/etc/security/limits.conf` as root, and adding (or altering) the line:

```
sgc_user soft nproc 4096
```

It may also be necessary to increase the hard limit:

```
sgc_user hard nproc 4096
```

4.4 SCTP tuning

For optimal performance, tune these kernel parameters:

Parameter	Recommended value	What it specifies
<code>net.core.rmem_default</code>	512000	Default receive buffer size (in bytes)
<code>net.core.wmem_default</code>	512000	Default send buffer size (in bytes)
<code>net.core.rmem_max</code>	2048000	Maximum receive buffer size (in bytes) This value limits the <code>so-rcvbuf</code> parameter. For details, please see Listening for and establishing SCTP associations on page 101 .
<code>net.core.wmem_max</code>	2048000	Maximum send buffer size (in bytes) This value limits the <code>so-sndbuf</code> parameter. For details, please see Listening for and establishing SCTP associations on page 101 .
<code>net.sctp.rto_min</code>	$40 < \text{rto_min} < 100$	Minimum retransmission timeout (in ms) This should be greater than the <code>sack_timeout</code> of the remote SCTP endpoints.

net.sctp.sack_timeout	40 < sack_timeout < 100	Delayed acknowledgement (SACK) timeout (in ms) Should be lower than the retransmission timeout of the remote SCTP endpoints.
net.sctp.hb_interval	1000	SCTP heartbeat interval (in ms)



Kernel parameters can be changed

- at runtime using sysctl command; for example: `sysctl -w net.core.rmem_max=2000000`
- or set permanently in `/etc/sysctl.conf` file.

4.5 SGC Installation

To install OCSS7: [unpack and configure](#) on page 55 then [create additional nodes](#) on page 56 .

4.5.1 Unpack and configure



SGC_HOME

The following instructions use `SGC_HOME` to represent the path to the SGC installation directory.

To begin the SGC installation and create the first node:

1

Unpack the SGC archive file

Run:

```
unzip ocss7-package-VERSION.zip
```

(replacing `ocss7-package-VERSION.zip` with the correct file name)

This creates the distribution directory , `ocss7-X.X.X.X` , in the current working directory.

2	Make sure that the <code>JAVA_HOME</code> environment variable is set to the location of the Oracle JDK installation. (for JDK installation, see the Operating System documentation and/or JDK vendor documentation).
3	<p>Configure basic cluster / node information</p> <p>If your installation will use more than a single node in a SGC cluster, then:</p> <ul style="list-style-type: none"> • Customize the <code>ss7.instance</code> property in <code>SGC_HOME/config/SGC.properties</code> to a value unique among all other nodes in the SGC cluster. • Define a cluster name through the <code>hazelcast.group</code> property in <code>SGC_HOME/config/SGC.properties</code>. The value of this property must be the same for all nodes that form a particular cluster. <p>If you are planning to use more than one SGC cluster in the same local network then:</p> <ul style="list-style-type: none"> • Set the <code>hazelcast.group</code> property in <code>SGC_HOME/config/SGC.properties</code> to a value unique among all other clusters in the same local network.

4.5.2 Creating additional nodes

After installing the first SGC node in a cluster, you can add more nodes by either:

- copying the installation directory of an existing node, and changing the `ss7.instance` property in `SGC_HOME/config/SGC.properties` to a value unique among all the other nodes in the cluster.

or

- repeating the installation steps for another node,
- setting the `ss7.instance` property in `SGC_HOME/config/SGC.properties` to a value unique among all other nodes in the cluster,
- setting the `hazelcast.group` in `SGC_HOME/config/SGC.properties` to the value chosen as cluster name, and
- repeating any other installation customization steps.

4.5.3 Layout of the SGC installation

A typical SGC installation contains these subdirectories:

Directory	Contents
.	(SGC installation directory) <ul style="list-style-type: none">main SGC Java Archive
bin	SGC management scripts
cli	command line interface installation, including start scripts, configuration, and logs
config	configuration files which may be edited by the user as required
doc	supplementary documentation included as a convenience, such as SNMP MIB files
lib	Java libraries used by the SGC
license	third-party software licenses
logs	log output from the SGC
var	persisted cluster configuration (<code>sgc.dat</code>) and temporary files used by the SGC management scripts

4.6 Running the SGC

4.6.1 SGC operations



JAVA_HOME

The SGC script expects the `JAVA_HOME` environment variable to be set up and point to a valid JVM version 7 or greater (expects executable file `JAVA_HOME/bin/java`).

The SGC is started and stopped using the `SGC_HOME/bin/sgc script`.

The `sgc` script runs SGC under a watchdog: if the SGC process exits for an unrecognized reason it is automatically restarted. Output from the SGC and from the watchdog script is redirected into a startup log file. The startup log files are in `SGC_HOME/logs` directory and are named `startup.<startup-time>`. If startup fails for any reason, details about the failure should be available in the startup file.

The `sgc` script is configured in `SGC_HOME/config/sgcenv`. The `sgcenv` file contains JVM parameters which cannot be provided in the `SGC.properties` file.

The `sgc` script can be run with the following arguments:

Command argument	Optional arguments	Description
start	<code>--nowait --jmxhost <host> --jmxport <port></code>	Starts the SGC using the configuration from <code>SGC_HOME/config/sgcenv</code> . <ul style="list-style-type: none"> The <code>--nowait</code> argument specifies that the startup script does not verify if SGC has started successfully, but just initializes <code>start</code> and exits. The <code>--jmxhost</code> and <code>--jmxport</code> arguments allow specifying different JMX listening sockets than the one defined in <code>SGC_HOME/config/sgcenv</code>.
stop	<code>--immediate</code>	Stops the SGC. Without the <code>--immediate</code> argument, a graceful shutdown is attempted. With <code>--immediate</code> , processes are killed.
restart	<code>--nowait --jmxhost <host> --jmxport <port> --immediate</code>	Equivalent of <code>stop</code> , then <code>start</code> .
test	<code>--jmxhost <host> --jmxport <port></code>	Runs the SGC in test mode. In test mode, SGC runs in the foreground; and logging is configured in <code>log4j.test.xml</code> , which prints more information to the console.
foreground	<code>--jmxhost <host> --jmxport <port></code>	Runs the SGC in foreground mode. SGC is not demonized.
status		Prints the status of SGC and returns one of these LSB-compatible exit codes: <ul style="list-style-type: none"> 0 = the SGC is running

- 1 = the SGC is dead and the `SGC_HOME/var/sgc.pid` file exists
- 3 = the SGC is not running

For example:

Start SGC	<pre>\$SGC_HOME/bin/sgc start \$SGC_HOME/bin/sgc start --nowait --jmxport 10111 --jmxhost 127.0.0.1</pre>
Stop SGC	<pre>\$SGC_HOME/bin/sgc stop \$SGC_HOME/bin/sgc stop --immediate</pre>
Check SGC status	<pre>\$SGC_HOME/bin/sgc status</pre>

4.6.2 Configuring `SGC_HOME/config/sgcenv`

The `SGC_HOME/config/sgcenv` file contains configuration parameters for the `sgc` script. The following settings are supported:

Variable name	Descriptions	Valid Values	Default
<code>JAVA_HOME</code>	Location of the JVM home directory.		
<code>JMX_HOST</code>	Host that SGC should bind to in order to listen for incoming JMX connections.	IPv4 or IPv6 address	127.0.0.1
<code>JMX_PORT</code>	<p>Port where SGC binds for incoming JMX connections.</p> <p>It is not recommended to use a port in the ephemeral range as these are used for short-lived TCP connections and may result in the SGC failing to start if the port is in use by another application. The ephemeral port range may be queried with <code>cat /proc/sys/net/ipv4/ip_local_port_range</code>.</p>	1 - 65535	10111

JMX_SECURE	<p>Whether or not the JMX connection should be secured with SSL/TLS.</p> <p>For details, please see Securing the SGC JMX management connection with SSL/TLS on page 74 .</p>	true , false	false
JMX_NEED_CLIENT_AUTH	<p>Whether or not the SGC should require a trusted client certificate for an SSL/TLS-secured JMX connection.</p> <p>For details, please see Securing the SGC JMX management connection with SSL/TLS on page 74 .</p>		
JMX_SECURE_CFG_FILE	<p>Path to the configuration file with properties used to secure the JMX management connection.</p> <p>For details, please see Securing the SGC JMX management connection with SSL/TLS on page 74 .</p>		
DEFAULT_STORE_PASSWORD	<p>Password used during generation of the key store and trust store used to secure the JMX management connection.</p> <p>For details, please see Securing the SGC JMX management connection with SSL/TLS on page 74 .</p>		changeit
MAX_HEAP_SIZE	Maximum size of the JVM heap space.		
MIN_HEAP_SIZE	Initial size of the JVM heap space.		
MAX_PERM_SIZE	Maximum size of the JVM permgen space.		
GCOPTIONS	Full override of default garbage collections settings.		
JVM_PROPS	Additional JVM parameters.		

	Modifications should add to the existing <code>JVM_PROPS</code> rather than overriding it. For example: <code>JVM_PROPS="\$JVM_PROPS -Dsome_option"</code> .		
LOGGING_CONFIG	The log4j configuration file to be used in normal mode (start/restart/foreground).		
LOGGING_TEST_CONFIG	The log4j configuration file to be used in test mode.		
SGC_CFG_FILE	Location of the SGC properties file.		
RUNONCE	Whether or not the watchdog is enabled. Disabling the watchdog may be required if the SGC is run under the control of some other HA systems.	0 = watchdog is enabled 1 = start SGC in background without watchdog script	
DEBUG	Enables additional script information.	0 = no additional debug information 1 = additional information enabled	
PREFER_IPV4	Prefers using IPv4 protocol. Set value to <code>false</code> to enable IPv6 support.	true = use only IPv4 false = use both IPv4 and IPv6	true
NUMAZONES	On NUMA architecture machines, this parameter allows selecting specific CPU and memory bindings for SGC.		
CPUS	On non-NUMA architecture machines, this parameter may be used to set SGC affinity to specific CPUs.		



`SGC_HOME/config/sgcenv` contains additional variables used to configure a secure JMX management connection. For details, please see [Securing the SGC JMX management connection with SSL/TLS](#) on page 74 .

4.7 Installing SGC as a service

To install SGC as a service, perform the following operations as user `root` :

1	<p>Copy <code>SGC_HOME/bin/sgcd</code> to <code>/etc/init.d</code> .</p> <pre># copy \$SGC_HOME/bin/sgcd /etc/init.d</pre>
2	<p>Grant execute permissions to <code>/etc/init.d/sgcd</code> :</p> <pre># chmod a+x /etc/init.d/sgcd</pre>
3	<p>Create the file <code>/etc/sysconfig/sgcd</code> . Assuming that the SGC is installed in <code>/opt/sgc/current</code> as user <code>sgc</code> , the file would have the following content:</p> <pre>SGC=/opt/sgc/current/bin/sgc SGCUSER=sgc</pre>
4	<p>Activate the service using the standard RedHat command:</p> <pre># chkconfig --add sgcd</pre>

5 Network Architecture Planning



Ensure you are familiar with the with the OCSS7 architecture before going further. In particular, ensure you have read the following:

- [Architectural Overview](#) on page
- [Configuring the SS7 SGC Stack](#) on page 81
- [General Configuration](#) on page 93
- [M3UA Configuration](#) on page 97
- [SGC TCAP Stack](#) on page 191

5.1 Network planning

When planning OCSS7 deployment, OpenCloud recommends preparing IP subnets that logically separate different kinds of traffic:

Subnet	Description
SS7 network	dedicated for incoming/outgoing SIGTRAN traffic; should provide access to the operator's SS7 network
SGC interconnect network	internal SGC cluster network with failover support (provided by interface bonding mechanism); used by Hazelcast and communication switch
Rhino traffic network	used for traffic exchanged between SGC and Rhino nodes
Management network	dedicated for managing tools and interfaces (JMX, HTTP)

5.2 SGC Stack network communication overview



The [SS7 SGC](#) on page uses multiple logical communication channels that can be separated into two broad categories:


- [SGC directly managed connections](#) on page 64 — connections established directly by SGC subsystems, configured as part of the SGC cluster-managed configuration
- [Hazelcast managed connections](#) on page 66 — connections established by Hazelcast, configured as part of static SGC instance configuration.

5.2.1 SGC directly managed connections

The following table describes network connections managed directly by the SGC configuration.

Protocol	Subsystem	Subnet	Defined by	Usage
TCP	TCAP	Rhino traffic network	stack-http-address and stack-http-port attributes of the node configuration object	Used in the first phase of communication establishment between the TCAP Stack (CGIN RA) and the SGC cluster. The communication channel is established during startup of the TCAP Stack (CGIN RA activation), and closed after a single HTTP request / response.
			stack-data-address and stack-data-port attributes of the node configuration object	Used in the second phase of communication establishment between the TCAP Stack (CGIN RA) and the SGC cluster. The communication channel is established and kept open until either the SGC Node or the TCAP Stack (CGIN RA) is shutdown (deactivated). This connection is used to exchange TCAP messages between the SGC Node and the TCAP Stack using a custom protocol. The level of expected traffic is directly related to the number of expected SCCP messages originated and destined for the SSN represented by the connected TCAP Stack.

				 When multiple TCAP Stacks representing the same SSN are connected to the SGC cluster, then message traffic is load-balanced between them.
	Communication Switch	SGC interconnect network	switch-local-address and switch-port attributes of the node configuration object	<p>Used by the communication switch (inter-node message transfer module) to exchange message traffic between nodes of the SGC cluster.</p> <p>The communication channel is established between nodes of the SGC cluster during startup, and kept open until the node is shut down. During startup, the node establishes connections to all other nodes that are already part of the SGC cluster. The level of expected traffic depends on the deployment model, and can vary anywhere between none and all traffic destined and originated by the SGC cluster.</p>  To avoid unnecessary resource use, minimise communication switch traffic when planning the deployment model. For details please see Inter-node message transfer on page 67 .
SCTP	M3UA	SS7 Network	local-endpoint and connection M3UA configuration objects	<p>Used by SGC nodes to exchange M3UA traffic with Signalling Gateways and/or Application Servers.</p> <p>The communication channel lifecycle depends directly on the SGC cluster configuration; that is, the enabled attribute of the connection configuration object and the state of the remote system with which SGC is to</p>

				communicate. The level of traffic should be assessed based on business requirements.
JMX over TCP	Configuration	Management network		<p>Used for managing the SGC cluster.</p> <p>Established by the management client Command-Line Management Console on page 173 , for the duration of the management session. The level of traffic is negligible.</p> <div>  <div> <p>See Also</p> <ul style="list-style-type: none"> • Installing the SGC on page 52 • Securing the SGC JMX management connection with SSL/TLS on page 74 </div> </div>

5.2.2 Hazelcast managed connections

Hazelcast uses a two-phase cluster-join procedure:

1. Discover other nodes that are part of the same cluster.
2. Establish one-to-one communication with each node found.

Depending on the configuration, the first step of the cluster-join procedure can be based either on UDP multicast or direct TCP connections. In the latter case, the Hazelcast configuration must contain the IP address of at least one other node in the cluster. Connections established in the second phase always use direct TCP connections established between all the nodes in the Hazelcast cluster.

Traffic exchanged over SGC interconnect network by Hazelcast connections is mainly related to:

- SGC runtime state changes
- SGC configuration state changes
- Hazelcast heartbeat messages.

During normal SGC cluster operation, the amount of traffic is negligible and consists mainly of messages distributing SGC statistics updates.

5.2.3 Inter-node message transfer

The communication switch (inter-node message transfer module) is responsible for transferring data traffic messages between nodes of the SGC cluster. After the initial handshake message exchange, the communication switch does not originate any network communication by itself. It is driven by requests of the TCAP or M3UA layers.

Usage of the communication switch involves additional message-processing overhead, consisting of:

- CPU processing time to encode and later decode the message — this overhead is negligible
- network latency to transfer the message between nodes of the SGC cluster — overhead depends on the type and layout of the physical network between communicating SGC nodes.

This overhead is unnecessary in normal SGC cluster operation, and can be avoided during deployment-model planning.

Below are outlines of scenarios involving communication switch usage: [Outgoing message inter-node transfer](#) on page 67 and [Incoming message inter-node transfer](#) on page 67 ; followed by tips for [Avoiding communication switch overhead](#) on page 68 .

Outgoing message inter-node transfer

A message that is originated by the TCAP stack (CGIN RA) is sent over the TCP-based data-transfer connection to the SGC node (node A). It is processed within that node up to the moment when actual bytes should be written to the SCTP connection, through which the required [DPC](#) on page is reachable. If the SCTP connection over which the DPC is reachable is established on a different SGC node (node B), then the communication switch is used. The outgoing message is transferred, using the communication switch, to the node where the SCTP connection is established (transferred from node A to node B). After the message is received on the destination node (node B) it is transferred over the locally established SCTP connection.

Incoming message inter-node transfer

A message received by an M3UA connection, with a remote Signalling Gateway or other Application Server, is processed within the SGC node where the connection is established (node A). If the processed message is a TCAP message addressed to a [SSN](#) on page available within the SGC cluster, the processing node is responsible for selection of a TCAP Stack (CGIN RA) corresponding to that SSN. The TCAP Stack (CGIN RA) selection process gives preference to TCAP Stacks (CGIN RAs) that are directly connected to the SGC node which is processing the incoming message. If a suitable locally connected TCAP Stack (CGIN RA) is not available, then a TCAP stack connected to another SGC node

(node B) in the SGC cluster is selected. After the selection process is finished, the incoming TCAP message is sent either directly to the TCAP Stack (locally connected TCAP Stack), or first transferred through the communication switch to the appropriate SGC node (transferred from node A to node B) and later sent by the receiving node (node B) to the TCAP Stack.



TCAP Stack (CGIN RA) selection

TCAP Stack selection is invoked for messages that start a new transaction (`TCAP BEGIN`) or are not a part of any transaction (`TCAP UNIDIRECTIONAL`). Messages that are a part of an existing transaction are directed to the TCAP Stack serving that transaction; that is, the TCAP Stack that received initial `TCAP BEGIN` message.

TCAP Stack selection is described by following algorithm:

1. Acquire a set of locally connected TCAP Stacks serving the appropriate SSN.
2. If the set of locally connected TCAP Stacks is empty, acquire a set of TCAP Stacks serving the appropriate SSN, cluster-wide.
3. Load balance incoming messages among the TCAP Stacks in the acquired set, in a round-robin fashion.

Avoiding communication switch overhead

A review of the preceding communication-switch usage scenarios suggests a set of rules for deployment, to help avoid communication-switch overhead during normal SGC cluster operation.

Scenario	Avoidance Rule	Configuration Recommendation
Incoming message inter-node transfer on page 67	If an SSN is available within the SGC cluster, at least one TCAP Stack serving that particular SSN must be connected to each SGC node in the cluster.	<p>The number of TCAP Stacks (CGIN RAs) serving a particular SSN should be at least the number of SGC nodes in the cluster.</p> <div> <p>Keep in mind that a single CGIN RA entity deployed within a Rhino cluster is instantiated on each Rhino node; this translates to the number of TCAP Stacks being equal to the number of Rhino nodes for each CGIN RA entity.</p> </div>

Outgoing message inter-node transfer on page 67	If the SGC Stack is to communicate with a remote PC (another node in the SS7 network), that PC must be reachable through an M3UA connection established locally on each node in the SGC cluster.	When configuring remote PC availability within the SGC Cluster, the PC must be reachable through at least one connection on each SGC node.
---	--	--

5.3 SGC cluster membership and split-brain scenario

The SS7 SGC Stack is a distributed system. It is designed to run across multiple computers connected across an IP network. The set of connected computers running SGC is known as a **cluster**. The SS7 SGC Stack cluster is managed as a single system image. SGC Stack clustering uses an **n-way, active-cluster architecture**, where all the nodes are fully active (as opposed to an active-standby design, which employs a live but inactive node that takes over if needed).

SGC cluster membership state is determined by Hazelcast based on network reachability of nodes in the cluster. Nodes can become isolated from each other if some networking failure causes a network segmentation. This carries the risk of a "split brain" scenario, where nodes on both sides of the segment act independently, assuming nodes on the other segment have failed. The responsibility of avoiding a split-brain scenario depends on the availability of a redundant network connection. For this reason, [network interface bonding](#) MUST be employed to serve connections established by Hazelcast.

Usage of a communication switch subsystem within the SGC cluster depends on the cluster membership state, which is managed by Hazelcast. Network connectivity as seen by the communication switch subsystem MUST be consistent with the cluster membership state managed by Hazelcast. To fulfil this requirement, the communication switch subsystem MUST be configured to use the same redundant network connection as Hazelcast.



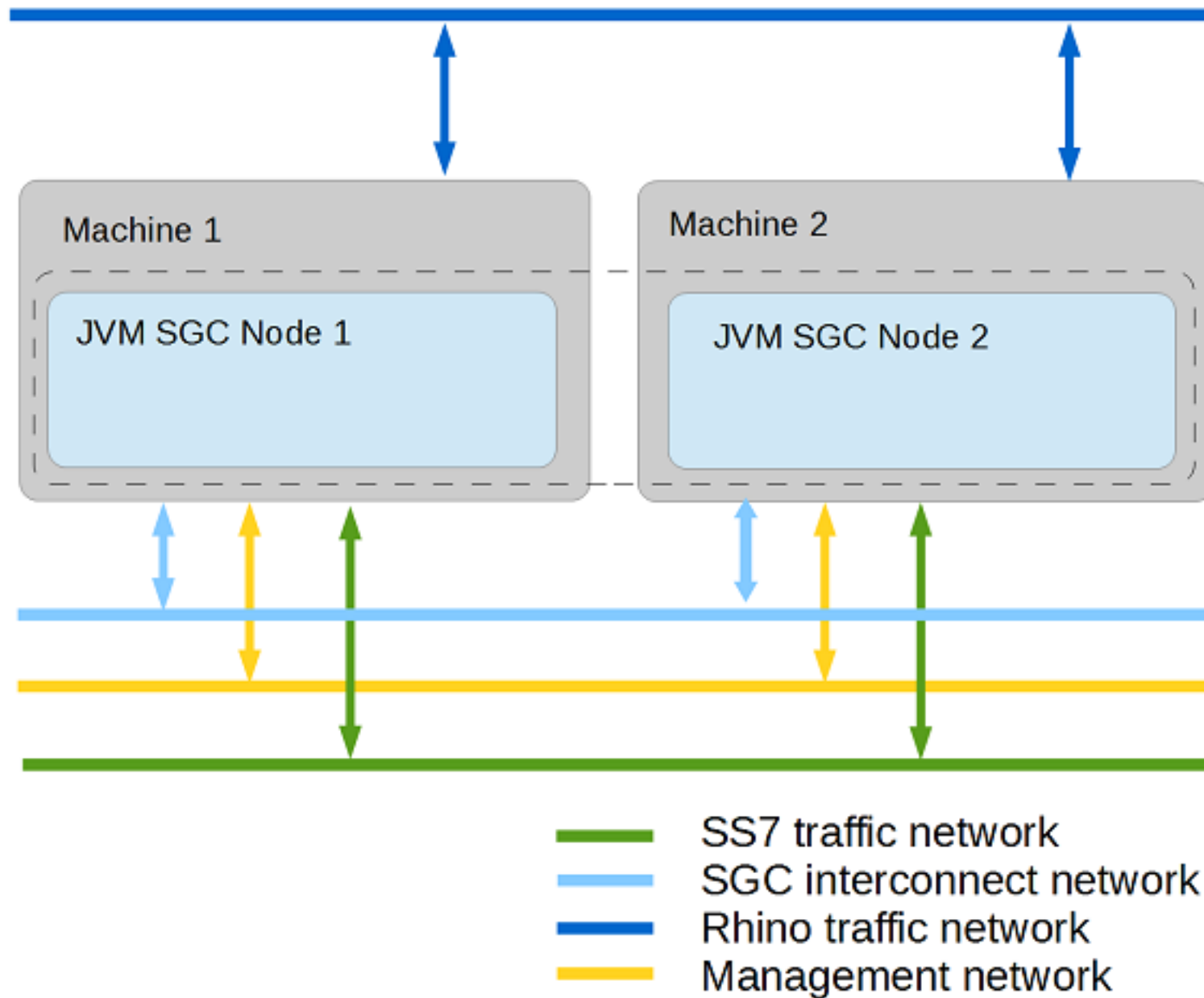
Network connection redundancy delivery method Both Hazelcast and the communication switch currently do not support network interface failover. This results in a requirement to use OS-level [network interface bonding](#) to provide a single logical network interface delivering redundant network connectivity.



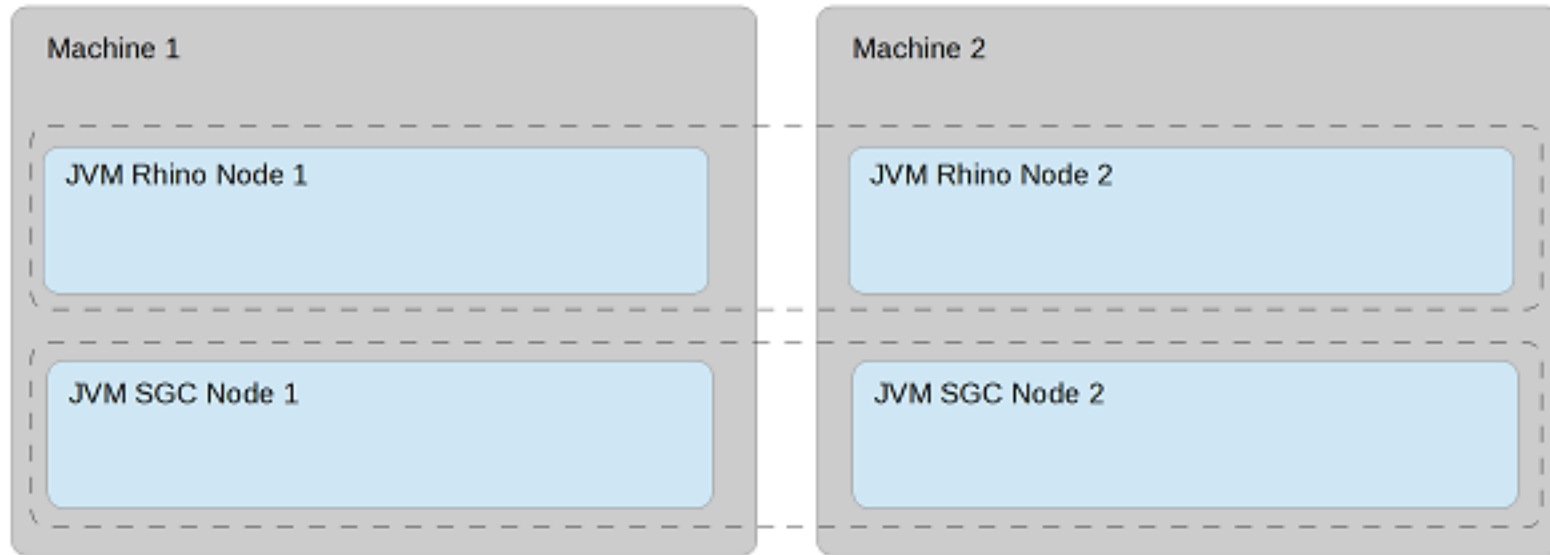
Network Path Redundancy The entire network path between nodes in the cluster must be redundant (including routers and switches).

5.4 Recommended physical deployment model

In order to take full advantage of the fault-tolerant and high-availability modes supported by the OC SS7 stack, OpenCloud recommends using at least two dedicated machines with multicore CPUs and two or more Network Interface Cards.



Each SGC node should be deployed on one dedicated machine. However hardware resources can be also shared with nodes of Rhino Application Server.



The OC SS7 stack also supports less complex deployment modes which can also satisfy high-availability requirements.



To avoid single points of failure at network and hardware levels, provide redundant connections for each kind of traffic. The SCTP protocol that SS7 traffic uses itself provides a mechanism for IP multi-homing. For other kinds of traffic, an interface-bonding mechanism should be provided. Below is an example assignment of different kinds of traffic among network interface cards on one physical machine.

	Network Interface Card 1	Network Interface Card 2
port 1	SS7 IP addr 1	SS7 IP addr 2
port 2	SGC Interconnect IP addr (bonded)	SGC Interconnect IP addr (bonded)

port 3	Rhino IP addr	
port 4	Management IP addr	



While not required, bonding Management and Rhino traffic connections can provide better reliability.

6 Securing the SGC JMX management connection with SSL/TLS

6.1 Default configuration of the JMX management connection

The default JMX configuration allows for unsecured JMX management connections from the local machine only. That is, the SGC SS7 stack by default listens for management connections on a local loopback interface. This allows for any JMX management client running on the same machine as the SGC stack instance to connect and manage that instance with no additional configuration.

6.2 Securing the JMX management connection with SSL/TLS



SGC_HOME SGC_HOME in the following instructions represents the path to the SGC Stack installation directory.

6.2.1 SGC stack secure configuration

The SGC SS7 stack can be configured to secure JMX management connections using the SSL/TLS protocol. The default installation package provides a helper shell script (`SGC_HOME/bin/sgckeygen`) that generates:

- `SGC_HOME/config/sgc-server.keystore` — a [JKS](#) repository of security certificates containing two entries: an SGC JMX server private key and a trust entry for the SGC JMX client certificate
- `SGC_HOME/config/sgc-client.keystore` — a JKS repository of security certificates containing two entries: an SGC JMX client private key and a trust entry for the SGC JMX server certificate
- `SGC_HOME/config/netssl.properties` — a Java properties file containing the configuration the SGC Stack uses during start-up (properties in this file point to the generated `sgc-server.keystore`)
- `SGC_HOME/config/sgc-trust.cert` — the SGC JMX server certificate, which can be imported to any pre-existing KeyStore to establish a trust relation.

To enable a secure JMX management connection:

1. Generate appropriate server / client private keys and certificates: run the `SGC_HOME/bin/sgckeygen` script .

2. Change the SGC stack configuration to enable the secure connection: edit the configuration file `SGC_HOME/config/sgcenv`, changing the `JMX_SECURE` variable value to `true`.



By default, the SGC stack is configured to require client authorization with a trusted client certificate. The straightforward approach is to use the generated `SGC_HOME/config/sgc-client.keystore` as part of the JMX management client configuration.



- For detailed information about creating a KeyStore, please see the Java Virtual Machine vendor documentation on the Oracle [JDK Tools and Utilities](#) page.
- For general information about SSL/TLS support, see the [JSSE Reference Guide](#).
- Configuration changes take effect on SGC stack instance restart.

6.2.2 Example client configuration for a JMX management secure connection

You can configure the JMX management connection from the [command line](#) on page 75 or using a [JDK tool](#) on page 75.

Configuring from the command line

To configure a secure JMX connection for the SGC Stack using a command-line management console, please see [Command-Line Management Console](#) on page 173.

Configuring with a generic JMX management tool

The Command-Line Management Console is a dedicated tool for operating and configuring the SGC stack; but there are many tools that support the JMX standard. Below are tips for letting them communicate with the SGC stack.

The SGC stack is equipped with scripts that enable JMX connector and provide a simple way to prepare all the necessary keys and certificates used during the SSL/TLS authentication process.



In order to connect to the SGC stack with an external tool, follow the tool's SGC stack secure configuration instructions.

For example, for Java VisualVM (part of the Sun/Oracle [JDK](#) on page) :

1. Generate the appropriate server / client private keys and certificates.

2. Copy the `SGC_HOME/config/sgc-client.keystore` to the machine where you want to start the Java VisualVM.
3. Start the Java VisualVM with parameters pointing to the relevant KeyStore file. For example: `jvisualvm -J-Djavax.net.ssl.keyStore=sgc-client.keystore -J-Djavax.net.ssl.keyStorePassword=changeit -J-Djavax.net.ssl.trustStore=sgc-client.keystore -J-Djavax.net.ssl.trustStorePassword=changeit`



The connection is secured only when using a remote/local JMX connector. Java VisualVM uses the "Attach API" to connect to locally running Java Virtual Machines, in effect bypassing the secure connection. In this case, client setup of a secure JMX connection is not required.

6.3 SGC stack JMX configuration properties

During SGC Stack instance startup, Java system properties are interrogated to derive configuration of the JMX RMI connector. Values of relevant properties can be configured using variables in the `SGC_HOME/config/sgcenv` configuration file.

6.3.1 Properties configurable using the `sgcenv` configuration file

The following JMX connector settings are supported in the `SGC_HOME/config/sgcenv` configuration file:

Variable	What it specifies	Values	Default
JMX_SECURE	whether to secure the JMX connection with SSL/TLS	true / false	false
JMX_NEED_CLIENT_AUTH	whether the SGC Stack requires a trusted client certificate for an SSL/TLS-secured JMX connection	true / false	true
JMX_SECURE_CFG_FILE	path to the configuration file with properties used to secure the JMX management connection		<code>SGC_HOME/config/netssl.properties</code>
DEFAULT_STORE_PASSWORD	password used to secure the KeyStore and TrustStore when generating them using the <code>SGC_HOME/bin/sgckeygen</code> script		changeit

The file specified by `JMX_SECURE_CFG_FILE` should be in the Java properties file format (as described in [Javadoc for Properties class](#)). Properties configurable using `JMX_SECURE_CFG_FILE` are related to the location and security of Java KeyStores containing the SGC stack private key, certificate, and trusted client certificate. Here are the properties configurable using `JMX_SECURE_CFG_FILE` :

Key	What it specifies
<code>javax.net.ssl.keyStore</code>	path to the Java KeyStore file containing the SGC Stack private key
<code>javax.net.ssl.keyStorePassword</code>	password protecting the KeyStore denoted by the <code>javax.net.ssl.keyStore</code> property
<code>javax.net.ssl.trustStore</code>	path to the Java KeyStore file containing the trusted client certificate
<code>javax.net.ssl.trustStorePassword</code>	password protecting the KeyStore denoted by the <code>javax.net.ssl.trustStore</code> property

6.3.2 Example `JMX_SECURE_CFG_FILE` properties file

The `JMX_SECURE_CFG FILE` generated by the `SGC_HOME/bin/sgckeygen` script looks like this:

```
#
This is a SSL configuration file.
#
and truststore location and password settings thus avoiding
#
to pass them as cleartext in the command-line.
javax.net.ssl.keyStore=./config/sgc-server.keystore
javax.net.ssl.keyStorePassword=changeit
javax.net.ssl.trustStore=./config/sgc-server.keystore
javax.net.ssl.trustStorePassword=changeit
```

6.4 SGC stack JMX connector configuration details

The details presented above should be sufficient to secure the SGC JMX management connection. However, for a customized solution (for example, using other start-up scripts), see the following JMX connector parameters supported by SGC stack.



Usually there is no need to customize the operation of the SGC stack JMX RMI connector, as relevant configuration is exposed through SGC start-up scripts.

Here are the Java system properties used to configure the SGC stack JMX RMI connector:

Key	What it specifies
	Values
<code>com.cts.ss7.management.jmxremote.host</code>	host that SGC should bind to in order to listen for incoming JMX connections
	resolvable host name or IP address (<code>0.0.0.0</code> to listen on all local interfaces)
<code>com.cts.ss7.management.jmxremote.port</code>	port where SGC binds for incoming JMX connections
	Valid port value <code>0 . . 65535</code> inclusive (<code>0</code> = system assigned)
<code>com.cts.ss7.management.jmxremote.ssl</code>	whether to enable secure monitoring using SSL (if false, then SSL is not used)
	true / false Default is false .
<code>com.cts.ss7.management.jmxremote.ssl.enabled.cipher.suites</code>	a comma-delimited list of SSL/TLS cipher suites to enable; used in conjunction with <code>com.cts.ss7.management.jmxremote.ssl</code>
	default SSL/TLS cipher suites

com.cts.ss7.management.jmxremote.ssl.enabled.protocols	a comma-delimited list of SSL/TLS protocol versions to enable; used in conjunction with com.cts.ss7.management.jmxremote.ssl
	default SSL/TLS protocol version
com.cts.ss7.management.jmxremote.ssl.need.client.auth	whether to perform client-based certificate authentication, if both this property and com.cts.ss7.management.jmxremote.ssl are true
	true / false Default is true .
com.cts.ss7.management.jmxremote.ssl.config.file	path to the configuration file with properties used to secure the JMX management connection (should be in Java properties file format)
	no default path (SGC_HOME/config/netssl.properties is assigned by the SGC start scripts)
javax.net.ssl.keyStore	KeyStore location *
	no default path
javax.net.ssl.keyStorePassword	KeyStore password *
	no default path
javax.net.ssl.trustStore	truststore location *
	no default path
javax.net.ssl.trustStorePassword truststore password	truststore password *
	no default path

* Can be defined in the `com.cts.ss7.management.jmxremote.ssl.config.file` configuration file

7 Configuring the SS7 SGC Stack

7.1 Configuration data

Configuration data can be separated into two main groups:

- **static** — configuration properties loaded from a file during SGC instance start-up, influencing the behaviour of that single instance
- **managed** — dynamic runtime configuration managed by and distributed within the SGC cluster.



SGC_HOME

In the following instructions, `SGC_HOME` represents the path to the SGC Stack installation directory.

7.1.1 Static SGC configuration

Static configuration is loaded during SGC instance startup; any configuration changes take effect after SGC Stack instance restart. Static configuration consists of:

- [static configuration properties](#) on page 81
- [Hazelcast configuration](#) on page 88
- [logging configuration](#) on page 90 .

Static SGC instance configuration

During SGC instance start-up, the `SGC_HOME/config/SGC.properties` configuration file is loaded.



The default configuration can usually be used without changes, except for two important properties:

- `ss7.instance` — an instance name (node name) that must be unique among instances in the same cluster
Later, when configuring the cluster, you are required to reference this node name.
- `hazelcast.group` — name of the cluster which this instance should join.

By default, the cluster name is generated; so each instance started with the default configuration will form a separate single-node cluster.

Here are the configuration properties available in SGC.properties:

Property	What it specifies	Default
<code>com.cts.ss7.commsp.heartbeatEnabled</code>	enables or disables the heartbeat timeout mechanism in the SGC. If this is enabled in the SGC then the TCAP stack must also be configured to send heartbeats, otherwise the connection between the TCAP stack and SGC will be marked as timed out after <code>com.cts.ss7.commsp.server.recvTimeout</code> seconds, resulting in an unstable TCAP stack to SGC connection. . See Data Connection Heartbeat Mechanism on page 204 for details	false
<code>com.cts.ss7.commsp.server.handshake.recvTimeout</code>	timeout (in seconds) waiting for a handshake between the SGC and TCAP stack	2
<code>com.cts.ss7.commsp.server.recvTimeout</code>	how many seconds to wait before the Peer closes the connection after not receiving anything Value must be greater than the heartbeat period configured in the TCAP stack on page 199 , see Data Connection Heartbeat Mechanism on page 204 for details.	11
<code>com.cts.ss7.commsp.server.sendQueueCapacity</code>	capacity of the Peer sending queue	1024

<code>com.cts.ss7.commsp.tcpNoDelay</code>	whether to disable the Nagle algorithm for the connection between the SGC and TCAP stack	true
<code>com.cts.ss7.commsw.client.letSystemChooseSourceAddress</code>	whether to ignore the configured switch-local-address when establishing a client intra-cluster-communication (comm switch module) connection; if <code>true</code> , the OS is responsible for choosing an appropriate source address <code>switch-local-address</code> is an attribute of the node configuration object	false
<code>com.cts.ss7.commsw.client.tcpNoDelay</code>	whether to disable the Nagle algorithm in intra-cluster communication client mode (comm switch module)	true
<code>com.cts.ss7.commsw.client.threads</code>	number of threads serving connections (client mode) from other SGC nodes; for intra-cluster communication (comm switch module) Each thread requires three File Descriptors. The recommended value should be one less than number of nodes in the cluster.	1
<code>com.cts.ss7.commsw.server.acceptor.threads</code>	number of threads accepting connections from other SGC nodes; for intra-cluster communication (comm switch module) Each thread requires three File Descriptors.	1
<code>com.cts.ss7.commsw.server.tcpNoDelay</code>	whether to disable the Nagle algorithm in intra-cluster communication server mode (comm switch module)	true

<code>com.cts.ss7.commsw.server.threads</code>	<p>number of threads serving connections (server mode) from other SGC nodes; for intra-cluster communication (comm switch module)</p> <p>Each thread requires three File Descriptors. The recommended value should be one less than number of nodes in the cluster.</p>	1
<code>com.cts.ss7.commsw.so_rcvbuf</code>	<p>size of the socket receive buffer (in bytes) used by intra-cluster communication (comm switch module)</p> <p>Use a value ≤ 0 for OS default.</p>	-1
<code>com.cts.ss7.commsw.so_sndbuf</code>	<p>size of the socket send buffer (in bytes) used by intra-cluster communication (comm switch module)</p> <p>Use a value ≤ 0 for OS default.</p>	-1
<code>com.cts.ss7.peer.data.acceptor.threads</code>	<p>number of threads accepting data connections from TCAP stack instances</p> <p>Each thread requires three File Descriptors.</p>	2
<code>com.cts.ss7.peer.data.service.threads</code>	<p>number of threads serving data connections from TCAP stack instances</p> <p>Each thread requires three File Descriptors. The recommended value should be equal to the number of TCAP stack instances served by the cluster.</p>	2
<code>com.cts.ss7.peer.http.acceptor.threads</code>	<p>number of threads accepting connections from TCAP stack instances requesting balancing information</p>	1

<code>com.cts.ss7.peer.http.service.threads</code>	number of threads serving accepted connections from TCAP stack instances requesting balancing information	1
<code>com.cts.ss7.shutdown.gracefulOnUncaughtEx</code>	whether to try a graceful shutdown first (value <code>true</code>) when shutting down because of a uncaught exception	<code>true</code>
<code>com.cts.ss7.shutdown.gracefulWait</code>	how long to wait (in milliseconds) during graceful shutdown, before forcing a shutdown	30000
<code>com.cts.ss7.shutdown.ignoreUncaughtEx</code>	whether an uncaught exception should result in instance shutdown (value <code>false</code>), or just be logged (value <code>true</code>) Severe errors (<code>java.lang.Error</code>) always result in shutdown without trying a graceful shutdown first.	<code>false</code>
<code>hazelcast.config.file</code>	optional path to the Hazelcast config file	<i>none</i>
<code>hazelcast.group</code>	cluster group to which this instance belongs	<code>STANDALONE_{random hexadecimal string}</code>
<code>sgc.alarming.factory.class</code>	implementation of the alarming factory	<code>com.cts.ss7.alarming.impl.AlarmingFactoryImpl</code>
<code>sgc.alarming.maxHistoryAge</code>	maximum alarming history age (in minutes)	1440
<code>sgc.data.dir</code>	property name used to get the path where the SGC data file should be stored	<i>current working directory at the time of startup</i>

<code>sgc.data.file</code>	property name used to get the actual SGC data file name	<code>sgc.dat</code>
<code>sgc.ignore_config_tampering</code>	whether the SGC should ignore validation of the MD5 signature of the XML configuration file	<code>false</code>
<code>sgc.ind.pool_size</code>	maximum number of inbound messages that may be processing or waiting to be processed	10000
<code>sgc.peer.pendingConnectionTTL</code>	<p>how long (in seconds) the SGC should consider a connection pending after peer node allocation, but before actual connection (after this time, the SGC assumes that the connection will not happen)</p> <p>This value is used by the load balancer to assist with balancing TCAP stacks amongst SGCs.</p> <p>If it is set too small then under certain failure conditions it may result in a TCAP stack continually trying to reconnect to an SGC data port that it cannot reach.</p> <p>The suggested value for this property is $(total_TCAP_stacks_connections * (total_SGCs_in_cluster - 1) + 1) * 4 + safety_factor$.</p> <p>The default value allows for 13 TCAP stacks and 2 SGCs under worst case failure conditions with a 4 second safety factor. The safety factor allows for network latency and timer precision.</p>	60
<code>sgc.req.pool_size</code>	maximum number of outbound messages that may be processing or waiting to be processed	10000

<code>sgc.tcap.maxPeers</code>	<p>maximum number of TCAP peers/clients connected to this SGC instance</p> <p>Each peer instance serving TCAP client connection consumes about 100MB of heap memory. In case of using large number of TCAP peers value of <code>MAX_HEAP_SIZE</code> parameter should be increased accordingly. For details, please see Configuring SGC_HOME/config/sgcenv on page 59 .</p>	5
<code>sgc.tcap.maxMigratedPrefixes</code>	<p>maximum number of migrated prefixes permitted to be handled by this SGC instance</p> <p>A value of 0 disables prefix migration to this SGC.</p> <p>Each migrated prefix consumes about 100MB of heap memory. Where a large number of migrated prefixes are required to be supported the value of <code>MAX_HEAP_SIZE</code> parameter should be increased accordingly. For details, please see Configuring SGC_HOME/config/sgcenv on page 59 .</p>	5
<code>sgc.worker.queue</code>	maximum number of tasks (messages to be processed) in one worker queue	256
<code>sgc.worker.threads</code>	number of threads used by the worker group to process inbound and outbound messages	32
<code>snmp.data.bc.file boot</code>	counter file name for snmp4j	<code>sgc-snmp-bc.dat</code>
<code>snmp.data.file</code>	file name for snmp4j persistent storage	<code>sgc-snmp.dat</code>

<code>ss7.http.tcp_no_delay</code>	whether TCAP manager uses a Nagle algorithm for incoming connections	false
<code>ss7.instance</code>	name of the SGC instance within the cluster	I1

7.1.2 Hazelcast cluster configuration

Hazelcast is a opensource In-Memory Data Grid. Hazelcast provides a set of distributed abstractions (such as data collections, locks, and task execution) that are used by subsystems of the SS7 SGC Stack to provide a single logical view of the entire cluster. SGC cluster membership state is directly based on Hazelcast cluster and node lifecycle.

The SGC Stack deployment package uses the default Hazelcast configuration, which is available in `SGC_HOME/config/hazelcast.xml.sample`.

Hazelcast configuration can be customized by providing a `hazelcast.xml` configuration file in the `config` subdirectory of the SGC Stack distribution (for example, by renaming `config/hazelcast.xml.sample` to `config/hazelcast.xml`). For a description of possible configuration options, and the format of the `hazelcast.xml` configuration file, please see the [Hazelcast In-Memory Data Grid Documentation](#) (Version 2.6).

The default Hazelcast configuration used by the SGC includes customisation of some hazelcast default values to support rapid cluster member failure detection and faster cluster merges following failure recovery. This configuration can be further refined as necessary.

Property	What it specifies	Default
<code>hazelcast.heartbeat.interval.seconds</code>	How frequently the hazelcast heartbeat algorithm is run.	1s
<code>hazelcast.max.no.heartbeat.seconds</code>	How long to wait before considering a remote hazelcast peer unreachable. If this value is set too small, then there is an increased risk of very short network outages or extended Java garbage collection triggering heartbeat failure detection. If this value is set too large, then there is an increased risk of SGC features becoming temporarily	5s

	<p>unresponsive due to blocking on necessary cluster-wide operations.</p> <p>It is important to balance the need to rapidly detect genuinely failed nodes with the need to protect against unnecessarily splitting and reforming the cluster as the split and merge operation is not instant and some SGC features may be temporarily unavailable during this process.</p>	
<code>hazelcast.merge.first.run.delay.seconds</code>	How long hazelcast will wait to attempt a cluster merge immediately following a node failure.	10s
<code>hazelcast.merge.next.run.delay.seconds</code>	How long hazelcast will wait to attempt a cluster merge following a node failure after the first merge attempt.	10s

The hazelcast heartbeat mechanism is used to detect cluster member failures; either network failures between members or actual process failures.

The default Hazelcast configuration used by the SGC is optimized for two cluster members. In the case where a larger cluster is required, it is strongly recommend that the "backup-count" parameter is configured to the total number of cluster members, minus one. This provides maximum resiliency in the case where more than one node may fail or be split from the cluster simultaneously.

There are two locations within hazelcast.xml where this parameter must be configured: under `<queue name="default">` and again under `<map name="default">` . Both of these must be configured for correct behaviour.

For example, in a three node cluster:

```
<queue name="default">
<max-size-per-jvm>10000</max-size-per-jvm>
<backup-count>2</backup-count>
...
<map name="default">
```

```
<backup-count>2</backup-count>
...
```

7.1.3 Logging configuration



For a description of the logging subsystem, please see [Logging](#) on page 159 .

7.2 Managed SGC cluster configuration

The SS7 SGC Stack is built around a configuration subsystem that plays a central role in managing the lifecycle of all configuration objects in SGC. This is called "managed configuration" (when the configuration subsystem manages the configuration data). During normal cluster operation, configuration data and management state is shared between all cluster nodes. Each cluster node persists configuration data in the local file system (as an XML file).

Managed configuration can be divided based on its cluster or node-level applicability:

- per node — this configuration is stored cluster-wide but relevant only for a given node (it references just the particular node for which it is relevant; for example, different SCTP associations may be created on specific nodes).
- cluster wide — this configuration is relevant and the same for each node (general configuration parameters, SCCP configuration, parts of M3UA)

Configuration is represented as a set of configuration objects. These configuration objects are managed through a set of [CRUD](#) on page commands exposed by the Command-Line Management Console distributed with the SGC SS7 Stack.



See also [Command-Line Management Console](#) on page 173

7.2.1 Configuration objects

Each configuration object within the SS7 SGC Stack is an instance of a particular configuration object type. The type of configuration object defines a set of attributes. For example, configuration objects of type `connection` are defined by attributes such as `port` , `conn-type` , and

others. A particular configuration object is identified by its `oname` attribute, which must be unique among all other configuration objects of that type.

7.2.2 Common configuration object attributes

These attributes are common to some or all configuration objects.

oname	<p>Every configuration object has an <code>oname</code> attribute that specifies its Object Name. It is an id that must be unique among all other objects of that particular type.</p> <p>Whenever an attribute is a reference to a configuration object, its value must be equal to the <code>oname</code> attribute of that referenced configuration object.</p>
dependencies	<p>A configuration object depends on another configuration object when any of its attributes reference that other configuration object. That is, the attribute value is equal to the <code>oname</code> attribute of the other configuration object. The configuration system keeps track of such references, and provides the <code>dependencies</code> attribute, which is a counter of how many other configuration objects depend on the current one.</p> <p>If the <code>dependencies</code> value is greater than 0 , the configuration object cannot be removed (to remove it, first all other configuration objects that depend on it must be removed).</p>
enabled	<p>Some configuration objects must be enabled before the configuration layer changes the related runtime state. All such objects expose the <code>enabled</code> attribute with values of <code>true</code> or <code>false</code> .</p>
active	<p>Some configuration objects with the <code>enabled</code> attribute also expose the <code>active</code> attribute, which tells if this object was successfully instantiated and is used in processing (for example if a connection is established). Possible values are <code>true</code> or <code>false</code> .</p>



See also

- [General Configuration](#) on page 93
- [M3UA Configuration](#) on page 97
- [SCCP Configuration](#) on page 107

- [SNMP Configuration](#) on page 114
- [Configuration Procedure](#) on page 122
- [Configuration Subsystem Details](#) on page 126

8 General Configuration

Below are attributes for configuring an [SGC Stack instance](#) on page 93 and the [entire cluster](#) on page 95 .



Attribute modification restrictions

- Only attributes that are "Reconfigurable" can be modified after a configuration object is created.
- Attributes that do not support "Active Reconfiguration" can be changed only when the configuration object is disabled (the value of its `disabled` attribute is `false`).

8.1 node

The `node` configuration object is for configuring an SGC Stack instance. Every SGC instance that is to be a part of cluster must be represented by a node configuration object. During startup, the SGC instance property `ss7.instance` is matched against the `oname` of existing nodes. If a match is found, the SGC instance will connect to the cluster, acting as that matching node.

Attribute name	Attribute description	Default
<code>oname</code>	object name	
<code>dependencies</code> Read only	number of items which depend on this object	
<code>enabled</code> Reconfigurable / Active reconfiguration	is object enabled	<code>false</code>
<code>active</code> Read only	is object active	

switch-local-address Reconfigurable	local address for CommSwitch on page to bind to	
switch-port Reconfigurable	local port for CommSwitch on page to bind to	9701
stack-data-address Reconfigurable	interface where stack can connect for data connection The TCAP Stack that is bundled with CGIN RA will use this connection to originate and receive TCAP messages.	value of stack-http-address if defined, otherwise value of switch-local-address
stack-data-port Reconfigurable	port where stack can connect for data connection The TCAP Stack that is bundled with CGIN RA will use this connection to originate and receive TCAP messages.	9703
stack-http-address Reconfigurable	interface where stack can get balancing information The TCAP Stack that is bundled with CGIN RA will use this connection to to register with the SGC node.	value of stack-data-address if defined, otherwise value of switch-local-address
stack-http-port Reconfigurable	port where stack can get balancing information The TCAP Stack that is bundled with CGIN RA will use this connection to to register with the SGC node.	9702
local-configuration Read only	node configuration Specifies values of properties that can be defined in the SGC.properties file.	



The JMX Object name for node is `SGC:type=general,category=node`

8.2 parameters

The `parameters` category specifies cluster-wide configuration (for a single cluster).

Attribute name	Attribute description	Default
<code>class1-resend-delay</code> Reconfigurable	delay (in milliseconds) before a class1 message is re-sent when route failure is detected	1000
<code>sp</code> Reconfigurable	ocal signalling point; a 14-bit integer; supported formats: simple integer (4106) or int(3bit)-int(8bit)-int(3bit) (2-1-2)	0
<code>sp-state</code> Reconfigurable	local signalling point state (OK / RESTRICTED / DOWN / CONG0 / CONG1 / CONG2)	OK
<code>ni</code> Reconfigurable	network indicator (INTERNATIONAL / SPARE / NATIONAL / RESERVED)	INTERNATIONAL
<code>sccp-sst-interval</code> Reconfigurable	interval (in milliseconds) between sent SST messages	10000
<code>sccp-timer-td</code> Reconfigurable	decay timer (in milliseconds) for SCCP congestion procedure (Q.714 section 5.2.8 SCCP management congestion reports procedure)	10000
<code>sccp-timer-ta</code> Reconfigurable	attack timer (in milliseconds) for SCCP congestion procedure (Q.714 section 5.2.8 SCCP management congestion reports procedure)	600

sccp-max-rsl-m Reconfigurable	maximum restriction sublevel per restriction level (Q.714 section 5.2.8 SCCP management congestion reports procedure)	4
sccp-max-rl-m Reconfigurable	maximum restriction level for each affected SP (Q.714 section 5.2.8 SCCP management congestion reports procedure)	8
sccp-timer-tconn Reconfigurable	timer (in milliseconds) for congestion abatement	10000
ntfy-m3ua-err Reconfigurable	time interval (in seconds) between successive notifications sent when the M3Ua layer is unable to process incoming messages	600
ntfy-sccp-err Reconfigurable	time interval (in seconds) between successive notifications sent when the SCCP layer is unable to process incoming messages	60
ntfy-tcap-err Reconfigurable	time interval (in seconds) between successive notifications sent when the TCAP layer is unable to process incoming messages	60
sccp-timer-reassembly Reconfigurable	reassembly timeout for SCCP reassembly procedure in milliseconds	2000
sccp-max-reassembly-processes Reconfigurable	maximum number of concurrent SCCP reassembly processes that may be active. 0 = reassembly not supported.	10000



The JMX Object name for parameters is `SGC:type=general,category=parameters`.

9 M3UA Configuration

The SS7 SGC Stack acts as one or more Application Servers when connected to the Signalling Gateway or another Application Server (in [IPSP](#) mode). M3UA configuration can be separated in two related domains:

- [Application Server and routes](#) on page 97
- [Listening for and establishing SCTP associations](#) on page 101



Attribute modification restrictions

Only attributes that are "Reconfigurable" can be modified after a configuration object is created. Attributes that do not support "Active Reconfiguration" can be changed only when the configuration object is disabled (the value of its `disabled` attribute is `false`).

9.1 Application Server and routes

After an SCTP association with SG is established, the as-connection that maps connections to Application Servers is checked. This allows for the SGC to inform SG which routing contexts should be active for that SCTP association. Also, with mapping internally, SGC decides which Point Codes are reachable through the SCTP association (the chain of dependencies is: [connection](#) on page 104 - [as-connection](#) on page 100 - [as](#) on page 97 - [route](#) on page 98 - [pc](#) on page 98).

9.1.1 as

Represents an Application Server — a logical entity serving a specific Routing Key. Defines the Application Server that the SGC Stack will represent after the connection to SG/IPSP peer is established.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only

enabled	is object enabled	false	Reconfigurable using active reconfiguration
traffic-maintenance-role	what kind of AS this is; whether it should handle sending (ACTIVE) or reception (PASSIVE) of ASP_(IN)ACTIVE	ACTIVE	
rc	routing context NOTE: This attribute is optional, but only one as with an unspecified rc can be used per association.		
state	State of the AS (DOWN , PENDING , ACTIVE , INACTIVE).		Read-only
pending-size	maximum number of pending messages, per node Applicable only to AS mode.	0	Reconfigurable
pending-time	maximum pending time (in milliseconds) Applicable only to AS mode.	0	Reconfigurable



The JMX Object name for as is `SGC:type=m3ua,category=as`.

9.1.2 route

Mapping of DPCs ([dpc](#) on page 99) accessible when a particular Application Server ([as](#) on page 97) is active.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
dpc-name	reference to DPC		

as-name	reference to AS		
priority	priority of the route (larger value is higher priority)	0	



The JMX Object name for route is `SGC:type=m3ua,category=route`.

9.1.3 dpc

A DPC defines a remote point code which is accessible from this SGC cluster. DPCs are used to define routes ([route](#) on page 98) which bind DPCs to specific Applications Server ([as](#) on page 97) definitions and SCTP associations ([connections](#) on page 98).

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
dpc	destination point code; a 14-bit integer; supported formats: simple integer (4106) or int(3bit)-int(8bit)-int(3bit) (2-1-2)		Reconfigurable
na	network appearance Attribute is optional		Reconfigurable
mss	maximum user data length per segment to send to this destination, supported values: 0-247 (0=do not segment)	247	Reconfigurable
muuss	maximum unsegmented SCCP message size to send to this destination as a single unsegmented message, supported values: 160-254 inclusive	254	Reconfigurable

pu <code>dt</code>	SCCP message type that this destination prefers to receive when there is a choice available, supported values: UDT, XUDT	UDT	Reconfigurable
--------------------	--	-----	----------------



The JMX Object name for `dpc` is `SGC:type=m3ua,category=dpc`.

9.1.4 as-precond

Before the Application Server ([as](#) on page 97) becomes active, the SGC Stack may require certain TCAP Stacks representing some SSN (CGIN RAs) to register with SGC. The Application Server will be activated after ALL defined preconditions (`as-precond`) are satisfied.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only
<code>as-name</code>	affected AS name		
<code>ssn</code>	subsystem number which must be connected		



The JMX Object name for `as-precond` is `SGC:type=m3ua,category=as-precond`.

9.1.5 as-connection

Mapping of Application Server ([as](#) on page 97) to SCTP association ([connection](#) on page 104), defining which Application Servers should be active on a particular connection.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		

dependencies	number of items which depend on this object		Read-only
as-name	reference to AS		
conn-name	reference to SGConnection		
daud-on-asp-ac	sending DAUD on asp-active	true	Reconfigurable



The JMX Object name for `as-connection` is `SGC:type=m3ua,category=as-connection`.

9.2 Listening for and establishing SCTP associations

Below are attributes for configuration directly related to establishing SCTP associations to M3UA peers.

9.2.1 local-endpoint

`local-endpoint`, together with `local-endpoint-ip` on page 103, defines the IP address where the SGC Node should listen for incoming SCTP associations. `local-endpoint` configuration is also used as the source address for outgoing SCTP associations. `local-endpoint` by itself defines the port and M3UA configuration for all connections that are associated with it.



Each SGC Node can use multiple local endpoints.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only

enabled	is object enabled	false	Reconfigurable using active reconfiguration
active	is object active		Read-only
node	SGC Node where the object is used		
port	local SCTP port		Reconfigurable
max-in-streams	maximum number of input streams requested by the local endpoint during association initialization	OS default	Reconfigurable
max-out-streams	maximum number of output streams requested by the local endpoint during association initialization	OS default	Reconfigurable
so-sndbuf	size of the socket send buffer	OS default	Reconfigurable
so-rcvbuf	size of the socket receive buffer	OS default	Reconfigurable
max-asp-active- count	maximum asp-active sending count	100	Reconfigurable
max-asp-in active-count	maximum asp-inactive sending count	100	Reconfigurable
max-asp-up- count	maximum asp-up sending count	100	Reconfigurable
max-asp-down- count	maximum asp-down sending count	100	Reconfigurable

no-delay	enables or disables a Nagle-like algorithm	true	Reconfigurable
so-linger	linger on close if data is present	OS default	Reconfigurable



The JMX Object name for `local-endpoint` is `SGC:type=m3ua,category=local-endpoint`.

9.2.2 local-endpoint-ip

Configuration of IPs for `local-endpoint` on page 101, to make use of SCTP multi-homed, feature-defined, multiple IPs for a single local endpoint.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
ip	IPv4 or IPv6 address		
local-endpoint-name	name of the referenced local endpoint		



The JMX Object name for `local-endpoint-ip` is `SGC:type=m3ua,category=local-endpoint-ip`.



IPv6 considerations

When using IPv6 addressing, remember to configure the `PREFER_IPV4` property in the `sgcenv` file. For details, please see [configuring SGC_HOME/config/sgcenv](#) on page 59

9.2.3 connection

`connection` , together with `conn-ip` on page 105 , defines the remote IP address of the SCTP association. The SGC Node will either try to establish that association (`CLIENT` mode) or expect a connection attempt from a remote peer (`SERVER` mode).

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only
<code>enabled</code>	is object enabled	false	Reconfigurable using active reconfiguration
<code>active</code>	is object active		Read-only
<code>port</code>	remote SCTP port		Reconfigurable
<code>local-endpoint-name</code>	reference name to local endpoint		
<code>conn-type</code>	specifies if acts as server for SCTP connect (<code>CLIENT/SERVER</code>)		Reconfigurable
<code>t-ack</code>	specifies how often ASP attempts to send <code>AspUP</code>	2	Reconfigurable using active reconfiguration
<code>t-daud</code>	specifies how often ASP attempts to send <code>DAUD</code>	60	Reconfigurable using active reconfiguration

t-reconnect	specifies time interval (in seconds) between connection attempts	6	Reconfigurable using active reconfiguration
state-maintenance-role	Specifies if SGC on this associations sends ASP-UP (ACTIVE) or expects to receive ASP-UP (PASSIVE)	ACTIVE	Reconfigurable
asp-id	asp identifier Attribute is optional		Reconfigurable
is-ipsp	specifies whether connection works in IPSP mode	true	
out-queue-size	maximum number of messages waiting to be written into the SCTP association	1000	Reconfigurable



The JMX Object name for connection is `SGC:type=m3ua,category=connection`.

9.2.4 conn-ip

Configuration of IPs for [connection](#) on page 104 , to make use of SCTP multi-homed, feature-defined,multiple IPs for a single connection.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
ip	IPv4 or IPv6 address		
conn-name	name of the referenced connection		



The JMX Object name for `conn-ip` is `SGC:type=m3ua,category=conn-ip`.



IPv6 considerations

When using IPv6 addressing, remember to configure the `PREFER_IPV4` property in the `SGC_HOME/config/sgcenv` file. For details, please see [configuring SGC_HOME/config/sgcenv](#) on page 59

10 SCCP Configuration

10.1 Global title translation



Any number of attributes (`gtencoding` , `trtype` , `natofaddr` , `numplan`) can be left empty to match any value in a [GT](#) on page . More specific rules, defining specific attribute values, are matched first.

To create a rule matching all GTs, also set `is-prefix = true` , and leave `addrinfo` empty.

To configure GT translation, see the following sections:

- [Incoming GT translation](#) on page 107
- [Outgoing GT translation](#) on page 109
- [Concerned Point Codes](#) on page 112
- [Load balancing](#) on page 113



Attribute modification restrictions

- Only attributes that are "Reconfigurable" can be modified after a configuration object is created.
- Attributes that do not support "Active Reconfiguration" can be changed only when the configuration object is disabled (the value of its `disabled` attribute is `false`).

10.2 Incoming GT translation

Use the following attributes to configure GT translation for incoming SCCP messages. Whenever the incoming called address for an SCCP message is a GT, it is matched against this configuration to determine whether it should be accepted and, optionally, which SSN should receive it.

10.2.1 inbound-gtt

Attribute name	Attribute description	Default	Modification
----------------	-----------------------	---------	--------------

oname	object name		
dependencies	number of items which depend on this object		Read-only
trtype	translation type Attribute optional; when unspecified matches ANY value		Reconfigurable
numplan	numbering plan Attribute optional; when unspecified matches ANY value		Reconfigurable
natofaddr	nature of address Attribute optional; when unspecified matches ANY value		Reconfigurable
addrinfo	address string Attribute optional; when unspecified and is-prefix is true , matches ANY value		Reconfigurable
is-prefix	specifies if address string contains prefix	false	Reconfigurable
ssn	local SSN that handles the traffic Attribute optional; when unspecified the SSN from the Called Party Address will be used		Reconfigurable



The JMX Object name for inbound-gtt is `SGC:type=sccp,category=inbound-gtt`.

10.3 Outgoing GT translation

GT translation configuration used for outgoing SCCP messages. Whenever the outgoing SCCP message called address is a GT, it is matched against this configuration to derive the destination PC and optionally SSN. After translation SCCP message called address may be modified according to the `replace-gt` on page 111 configuration.

10.3.1 outbound-gt

Use the following attributes to configure GT translation in an outgoing direction. An outgoing SCCP message that contains a GT in the called address parameter will be matched against the `outbound-gt` definitions.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only
<code>trtype</code>	translation type Attribute optional; when unspecified matches ANY value		Reconfigurable
<code>numplan</code>	numbering plan Attribute optional; when unspecified matches ANY value		Reconfigurable
<code>natofaddr</code>	nature of address Attribute optional; when unspecified matches ANY value		Reconfigurable
<code>addrinfo</code>	address string Attribute optional; when unspecified and <code>is-prefix</code> is true matches ANY value		Reconfigurable
<code>is-prefix</code>	specifies if the address string contains a prefix	false	Reconfigurable



The JMX Object name for `outbound-gt` is `SGC:type=sccp,category=outbound-gt`.

10.3.2 outbound-gtt

Use these attributes to represent a Destination Point Code where the SCCP message with a matching GT (referenced through the `gt` attribute) will be sent.



Multiple `outbound-gt` definitions can reference a single outbound GT. In such cases, a [load-balancing](#) on page 113 procedure is invoked.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only
<code>gt</code>	reference to O-GT		Reconfigurable
<code>dpc</code>	destination point code; a 14-bit integer; supported formats: simple integer (4106) or int(3bit)-int(8bit)-int(3bit) (2-1-2)		Reconfigurable
<code>replace-gt</code>	reference to <code>replace-gt</code>		Reconfigurable
<code>priority</code>	priority of this translation (larger value is higher priority)		Reconfigurable



The JMX Object name for `outbound-gtt` is `SGC:type=sccp,category=outbound-gtt`.

10.3.3 replace-gt

These attributes may be used to modify the SCCP message's called address parameter, after a matching GT is found through the use of [outbound-gt](#) on page 109 and [outbound-gtt](#) on page 110 .

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
route-on	what will be inserted in the SCCP called address; allowed values are: <ul style="list-style-type: none"> SSN — route on SSN; include translation-derived DPC GT — route on GT RI — leave intact. 	RI	Reconfigurable
gtencoding	new encoding of the address Attribute optional		Reconfigurable
trtype	new translation type Attribute optional		Reconfigurable
numplan	new numbering plan Attribute optional		Reconfigurable
natofaddr	new nature of address Attribute optional		Reconfigurable
addrinfo	new address string in hex/bcd format Attribute optional		Reconfigurable

ssn	specify new SSN to add to GT Attribute optional		Reconfigurable
gti	new global title indicator; allowed values are: 0 (no GT), 1 , 2 , 3 , or 4 Attribute optional		Reconfigurable



The JMX Object name for `replace-gt` is `SGC:type=sccp,category=replace-gt`.

10.4 Concerned Point Codes

10.4.1 cpc

CPC on page configuration stores information about remote SCCP nodes that should be informed about the local subsystem availability state.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
dpc	concerned point code which is notified about status changes; a 14-bit integer; supported formats: simple integer (4106) or int(3bit)-int(8bit)-int(3bit) (2-1-2)		



The JMX Object name for `cpc` is `SGC:type=sccp,category=cpc`.

10.5 Load balancing

Global Title translation may be used to provide load balancing and high availability functions. If there is more than one `outbound-gtt` on page 110 referencing a single `outbound-gt` on page 109 , the SCCP layer is responsible for routing the message to one of the available SCCP nodes (destination point codes). If the SCCP message is a subsequent message in stream class 1 (sequenced connectionless) and the previously selected SCCP node (PC) is still reachable, then that previously used PC is selected. If there is any other message for which GT translation results in multiple reachable point codes, messages are load balanced among the available PCs with highest priority.

The pseudo-algorithm is:

1. `outbound-gtt` s referencing the same GT (`outbound-gt`) are grouped according to their priority (larger value is higher priority).
2. Unreachable PCs are filtered out.
3. If there is more than one PC of highest priority, then messages are load balanced using a round robin algorithm between those PCs.

Whenever the `prefer-local` attribute of `outbound-gtt` is set to value `true` , routes local to the node are used in that algorithm (if such routes are currently available; otherwise `prefer-local` is ignored). Routes local to the node are those that are available through an SCTP association that was established by that particular node.

11 SNMP Configuration

11.1 Interoperability with SNMP-aware management clients

The SS7 SGC stack includes an [SNMP](#) agent, for interoperability with external SNMP-aware management clients. The SGC SNMP agent provides a read-only view of SGC statistics and alarms (through SNMP polling), and supports sending SNMP notifications related to SGC alarms and notifications to an external monitoring system.

In a clustered environment, individual SGC nodes may run their own instances of the SNMP agent, so that statistics and notifications can still be accessed in the event of node failure. Each node is capable of running multiple SNMP agents with different user-defined configuration.



For detailed information about SGC exposed statistics, please see [Statistics](#) on page 149 . For details about SGC alarms and notifications, please see [Alarms](#) on page 131 and [Notifications](#) on page 143 .



Attribute modification restrictions

- Only attributes that are "Reconfigurable" can be modified after a configuration object is created.
- Attributes that do not support "Active Reconfiguration" can be changed only when the configuration object is disabled (the value of its `disabled` attribute is `false`).

11.2 SNMP configuration

Each `snmp-node` configuration object represents an SNMP agent running as part of a particular SGC node. Exposed configuration allows a single SNMP agent to support a single version of the SNMP protocol. Currently supported SNMP versions are 2c and 3 . Multiple `snmp-node`s (SNMP agents) can run within a single SGC node. In a clustered environment, each newly created [snmp-node](#) on page 114 is automatically connected to the existing [target-address](#) on page 115 and [usm-user](#) on page 117 .

11.2.1 snmp-node

`snmp-node` represents an SNMP agent running as part of a particular SGC node.

Attribute name	Attribute description	Default	Modification
oname	object name		
dependencies	number of items which depend on this object		Read-only
enabled	is object enabled	false	Reconfigurable using active reconfiguration
active	is object active		Read-only
node	SGC node where the object is used		
port	local SNMP listening port		Reconfigurable
host	local SNMP listening bind address	127.0.0.1	Reconfigurable
community	community for read operations	public	Reconfigurable
snmp-version	SNMP version (supported values: v3 , v2c)	v2c	Reconfigurable



The JMX Object name for `snmp-node` is `SGC:type=snmp,category=snmp-node`.

11.2.2 target-address

`target-address` is cluster wide and represents an SNMP notification target (defines where SNMP notifications will be sent and which protocol version is used).

Attribute name	Attribute description	Default	Modification
oname	object name		

dependencies	number of items which depend on this object		Read-only
transportDomain	SNMP transport domain (SNMP transport protocol supported values: <code>TcpIpv4</code> , <code>UDP</code> , <code>UdpIpv6z</code> , <code>UdpIpv4z</code> , <code>TcpIpv6</code> , <code>TcpIpv4z</code> , <code>TcpIpv6z</code> , <code>UdpDns</code> , <code>UdpIpv6</code> , <code>UdpIpv4</code> , <code>TLSUDP</code> , <code>TcpDns</code> , <code>TLSTCP</code>)		Reconfigurable
target-host	target host address		Reconfigurable
target-port	target port	162	Reconfigurable
timeout	timeout value (in units of 0.01 seconds) after which unacknowledged SNMP notifications (type <code>inform</code>) will be retransmitted	200	Reconfigurable
retries	number of retransmissions of unacknowledged SNMP notifications (type <code>inform</code>)	1	Reconfigurable
community	community name definition	<code>public</code>	Reconfigurable
notifyType	SNMP notification mechanism (supported values: <code>trap</code> — asynchronous unacknowledged notification, <code>inform</code> — asynchronous acknowledged notification)	<code>trap</code>	Reconfigurable
snmp-version	SNMP version (supported values: <code>v3</code> , <code>v2c</code>)	<code>v2c</code>	Reconfigurable



The JMX Object name for `target-address` is `SGC:type=snmp,category=target-address` .



`target-address` can be created, deleted, and its attributes reconfigured, only when all referenced [snmp-node](#) on page 114 s are disabled.

11.2.3 usm-user

`usm-user` is cluster wide and represents the SNMP v3 [USM](#) on page user and authentication details.

Attribute name	Attribute description	Default	Modification
<code>oname</code>	object name		
<code>dependencies</code>	number of items which depend on this object		Read-only
<code>authProto</code>	authentication protocol (supported values: SHA , MD5 , NONE)	SHA	Reconfigurable
<code>authPassphrase</code>	authentication protocol pass phrase		Reconfigurable
<code>privProto</code>	privacy protocol (supported values: AES192 , DES , DES3 , AES256 , NONE , AES128)	DES	Reconfigurable
<code>privPassphrase</code>	privacy protocol pass phrase		Reconfigurable
<code>community</code>	specifies community	public	Reconfigurable



The JMX Object name for `usm-user` is `SGC:type=snmp,category=usm-user`.



`usm-user` can be created, deleted, and its attributes reconfigured only when all referenced [snmp-node](#) on page 114 s are disabled.

11.3 SGC Stack MIB definitions

MIB definitions for the SGC stack are separated into two files:

- [COMPUTARIS-MIB](#) — basic OID definitions used by the SGC stack
- [CTS-SGC-MIB](#) — SNMP managed objects and SNMP notifications used by the SGC stack.



MIB definitions are also included in the SGC Stack release package under `./doc/mibs/`

11.3.1 SNMP managed objects

Managed objects defined in CTS-SGC-MIB can be separated in two groups:

- [objects representing tables with SGC statistics](#) on page 118
- [objects representing tables with alarms and/or alarm history](#) on page 119 .

Statistics managed objects

Here are the managed objects representing SGC statistics:

Symbolic OID	Numerical OID	Equivalent Statistics on page 149 attribute
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcAssociationTable</code>	<code>.1.3.6.1.4.1.35787.1.1.1</code>	AssociationInfo on page 151
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcAsStateTable</code>	<code>.1.3.6.1.4.1.35787.1.1.2</code>	AsInfo on page 150
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcLocalSsnTable</code>	<code>.1.3.6.1.4.1.35787.1.1.3</code>	LocalSsnInfo on page 153

.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcTcapConnectionTable	.1.3.6.1.4.1.35787.1.1.4	TcapConnInfo on page 156
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcDpcStatusTable	.1.3.6.1.4.1.35787.1.1.5	DpcInfo on page 152
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcGtRoutingTable	.1.3.6.1.4.1.35787.1.1.6	OgtInfo on page 154
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcPcRoutingTable	.1.3.6.1.4.1.35787.1.1.7	PcInfo on page 155
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcRemoteSsnTable	.1.3.6.1.4.1.35787.1.1.8	RemoteSsnInfo on page 156
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcObjects.sgcHealthTable	.1.3.6.1.4.1.35787.1.1.9	HealthInfo on page 157

Alarms managed objects

Here are the managed objects representing SGC alarms:

Symbolic OID	Numerical OID	Equivalent Alarms on page 131 MBean
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcEvents.sgcActiveAlarmsTable	.1.3.6.1.4 .1.35787.1.2.5	alarm-table on page 167
.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObje cts.sgcEvents.sgcEventHistoryTable	.1.3.6.1.4 .1.35787.1.2.6	alarm-history on page 168

11.3.2 SNMP notifications

The MIB defined in CTS-SGC-MIB specifies two notification types that can be emitted by SGC:

- SGC Alarm — emitted whenever an SGC Alarm is raised
- SGC Notification — emitted whenever an SGC Notification is emitted.

Here are the notification types emitted by SGC:

Symbolic OID	Numerical OID
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcAlarm</code>	<code>.1.3.6.1.4.1.35787.1.2.2</code>
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcNotification</code>	<code>.1.3.6.1.4.1.35787.1.2.3</code>

Here is the content of SGC-emitted SNMP notifications:

Symbolic OID	Numerical OID	Details
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcAlarmObjects.sgcEventId</code>	<code>.1.3.6.1.4.1.35787.1.2.4.2</code>	unique SGC notification / alarm identifier
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcAlarmObjects.sgcAlarmName</code>	<code>.1.3.6.1.4.1.35787.1.2.4.4</code>	name of the SGC alarm / notification type
<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcAlarmObjects.sgcAlarmSeverity</code>	<code>.1.3.6.1.4.1.35787.1.2.4.5</code>	SGC alarm / notification severity

<code>.iso.org.dod.internet.private.enterprises. ctsRegMIB.ctsObjects.sgcEvents.sgcAlarmObjects. sgcAdditionalInfo</code>	<code>.1.3.6.1.4 .1.35787.1.2.4.</code>	comma-separated list of SGC Alarm / Notification type specific attribute= value pairs; this set of attributes depends on SGC Alarm / Notification type and is described in Alarm Types on page 133 and Notification Types on page 144
---	---	--

12 Configuration Procedure

This is a high-level description of the usual SGC configuration procedure listing the commands involved for each of the three layers:

- [general configuration](#) on page 122 ,
- [SCCP](#) on page 123 , and
- [M3UA](#) on page 124 .

It is up to the user to provide the details required by the commands; links to the configuration reference material for each command have been provided to make this easier.

12.1 General configuration

Step	Operation	
1	<p>Set the cluster-wide local SS7 Point Code address.</p> <p>Modify the sp attribute to the desired PC in ITU format.</p> <p>The ITU point code uses 14 bits. Supported formats are:</p> <ul style="list-style-type: none">• simple integer, such as 4106• 3bit-8bit-3bit format (3 most significant bits, 8 middle bits, 3 least significant bits) as decimal integer values, such as 2-1-2 .	modify-parameters on page 95
2	<p>Create two new nodes for the cluster.</p> <p>The oname node attribute must be equal to the <code>ss7.instance</code> property value of the SGC Stack instance that will act as that particular node.</p>	create-node on page 93

12.2 SCCP configuration

Below are the steps for configuring [outgoing](#) on page 123 and [incoming GTs](#) on page 123 to be translated, and [CPCs](#) on page 124 .



Configuring SCCP GT Translation and Concerned Point Codes is optional.

12.2.1 Outgoing GT

For each outgoing GT to be translated, repeat these steps:

Step	Operation	
1	Create the GT definition for outbound use.	create-outbound-gt on page 109
2	Create the address definition that should be the result of matching the previously defined GT. Be sure that the <code>gt</code> attribute is equal to the <code>oname</code> of the <code>outbound-gt</code> definition from the previous step.	create-outbound-gtt on page 110
3	Optionally, as a result of matching a particular GT, modify the called address before sending. After creating the <code>replace-gt</code> , be sure to modify the <code>replace-gt</code> attribute of the <code>outbound-gtt</code> created in the previous step	create-replace-gt on page 111

12.2.2 Incoming GT

For each incoming GT to be translated, repeat this step:

Step	Operation	
1	Create a GT and address definition (SSN) that should be the result of translation.	create-inbound-gtt on page 107
2	Create the GT definition for outbound use, making sure it matches the inbound GTT correctly.	create-outbound-gt on page 109

3	Create the address definition that should be the result of matching the previously defined GT. Be sure that the <code>gt</code> attribute is equal to the <code>oname</code> of the <code>outbound-gt</code> definition from the previous step.	create-outbound-gtt on page 110
---	--	---



The second and third commands may look somewhat surprising, as they create an outbound GTT rule. SCCP's service messages (UDTS and XUDTS) may be generated locally in response to traffic we are attempting to send, and these service messages are routed as *outbound* messages. It is safest to create outbound GTT rules mirroring your inbound GTT rules in case they are needed by your network configuration.

12.2.3 Concerned Point Code

For each remote signalling node that is concerned about local SSN availability, repeat this step:

Step	Operation	
1	Define the PC that should be notified about subsystem availability changes.	create-cpc on page 112

12.3 M3UA configuration

Below are instructions for configuring M3UA in [AS](#) on page , [IPSP Client](#) on page , and [IPSP Server](#) on page modes.



See Also

[RFC 4666 section 4.3. AS and ASP/IPSP State Maintenance](#)

Step	Operation	
1	If not done previously, define a <code>local-endpoint</code> for the existing node.	create-local-endpoint on page 101
2	Define IPs for the <code>local-endpoint</code> .	create-local-endpoint-ip on page 103

3	<p>If you are:</p> <ul style="list-style-type: none"> • connecting to an SG , set the: <ul style="list-style-type: none"> • <code>is-ipsp</code> attribute to <code>false</code> , • <code>conn_type</code> attribute to <code>CLIENT</code> , and • <code>state-maintenance-role</code> to <code>ACTIVE</code> . • using IPSP mode as a client , set the: <ul style="list-style-type: none"> • <code>is-ipsp</code> attribute to <code>true</code> , • <code>conn_type</code> attribute to <code>CLIENT</code> , and • <code>state-maintenance-role</code> to <code>ACTIVE</code> . • using IPSP mode as a server , set the: <ul style="list-style-type: none"> • <code>is-ipsp</code> attribute to <code>true</code> , • <code>conn_type</code> attribute to <code>SERVER</code> , and • <code>state-maintenance-role</code> to <code>PASSIVE</code> . 	create-connection on page 104
4	Define one or more IP addresses for connection.	create-conn-ip on page 105
5	<p>Define one or more Application Server (Routing Contexts).</p> <p>Set the <code>traffic-maintenance-role</code> attribute to <code>ACTIVE</code> .</p>	create-as on page 97
6	Define one or more Destination Point Codes that will be available for the AS.	create-dpc on page 99
7	Define one or more routes that associate previously created DPC(s) and AS(s).	create-route on page 98
8	Define one or more associations for AS(s) that are available through particular connection(s).	create-as-connection on page 100
9	Enable the <code>node</code> , <code>as</code> , <code>local-endpoint</code> , and <code>connection</code> .	

13 Configuration Subsystem Details

13.1 Stack configuration and cluster joining

The main functions of the configuration subsystem are:

- managing, distributing, and persisting SGC Stack configuration
- performing the cluster-join procedure.

Below are details of the [cluster-join procedure](#) on page 126 (which is part of SGC Node initialization) and a basic description of the [JMX MBeans](#) on page 127 exposed by the configuration subsystem that may be of use when developing custom management tools.



SGC_HOME

In the following instructions, `SGC_HOME` represents the path to the SGC Stack installation directory.

13.2 Cluster-join procedure

During startup, if the SGC cluster already exists, a node instance initiates a cluster-join procedure. The configuration system loads a locally persisted configuration and compares its [version vector](#) against the current cluster configuration.

IF	THEN
Versions are equal, or the local version is stale.	The SGC node uses the cluster configuration state. It joins the cluster and instantiates a cluster-wide and per-node state.
The local version is greater.	The local instance first instantiates all configuration objects which are not present in the cluster. Then it updates all configuration objects which are defined both in the cluster and in the local configuration. Finally, it deletes all configuration objects which are not present in the local configuration.

There is a version conflict.	The node aborts the cluster-join procedure, outputs a failure log message, and aborts start up.
------------------------------	---

13.2.1 Version conflict reconciliation

The local node stores a copy of the configuration in the `SGC_HOME/var/sgc.dat` file in the SS7 SGC node working directory. This is an XML file containing the entire cluster configuration as last known by that node. The usual approach to configuration reconciliation is for the node to join the cluster and use the current cluster configuration to initialize (dropping its local state).



To force this behaviour, remove or rename the `SGC_HOME/var/sgc.dat` file containing persisted configuration.



Configuration backup

After each configuration change a new version of `sgc.dat` file is created, before that current version is renamed to `SGC_HOME/var/sgc.dat.old`.

13.3 Factory MBeans for configuration objects

The SS7 SGC Stack is built around a configuration subsystem that plays a central role in managing the lifecycle of all configuration objects in SGC. SS7 SGC configuration is exposed through a set of JMX MBeans on each node. The configuration subsystem exposes a set of "factory" MBeans that are responsible for creating configuration objects of a certain type.

Each factory MBean exposes either one or two `create-` operations used to create new configuration objects (a `create` operation that accepts more parameters allows for defining optional configuration object attributes during creation). The creation of a configuration object results in the creation of an MBean representing the state of that configuration object. Attributes can be modified directly through that MBean, which also exposes a `remove` operation that allows removal of the configuration object (and associated MBeans).



Command-Line Management Console

These processes are abstracted away by the [Command-Line Management Console](#) on page 173 and exposed as a set of [CRUD](#) on page commands.

13.3.1 Configuration MBean naming

SGC Configuration MBean names use the common domain `SGC` and a set of properties:

- `type` — represents a subsystem / layer (`general` , `m3ua` , `sccp` , or `snmp`)
- `category` — represents the name of the factory in the subsystem
- `id` — represents that instance of the processing object.

For example, the cluster node factory MBean has the name `SGC:type=general,category=node` , which exposes the `create-node` operation. Invoking the `create-node` operation creates a processing object representing a node in the cluster with the object name `SGC:type=general,category=node,id=NODE_NAME` . The `id` property is set based on the `oname` (object name) parameter of the `create-node` operation.

Most GUI-based JMX-Management tools represent the naming space as a tree of MBean objects, like this:

MBeans Browser

MBeans

JMImplementation

SGC

general

node

1-1

1-2

parameters

m3ua

Attributes | Operations | Notifications | Metadata

MBeanInfo

Name	Value
MBeanInfo	
Info:	
ObjectName	SGC:type=general,category=node,id=1-1
ClassName	1-1
Description	Managed object NODE



There is a single special MBean object named `SGC:type=general,category=parameters` that is neither a factory MBean nor a processing object. This MBean stores cluster-wide system parameters.

14 Operational State and Instance Management

14.1 Operational state

[SS7 SGC](#) on page Stack operational state information is exposed cluster wide. That is, the same operational state information is exposed on each cluster node, independent of the particular source of information (particular node). It is enough to connect to a single cluster node to observe the operational state of all other nodes in the cluster.

The same operational state is exposed through:

- commands exposed by the Command-Line Management Console that is distributed with the SGC SS7 Stack
- the SNMP protocol when an SNMP agent is configured to run within a particular SGC node.



See also

- [Command-Line Management Console](#) on page 173
- [SNMP Configuration](#) on page 114



Notification propagation

An important distinction related to SS7 SGC Stack notification support is that notifications (both through JMX and SNMP) are sent only through the node that actually observes a related event (for example when a connection goes down).

Operational state is exposed through [Alarms](#) on page 131 , [Notifications](#) on page 143 , and [Statistics](#) on page 149 . Operating details of the SS7 SGC stack can be observed using the [Logging](#) on page 159 subsystem.

14.2 Static SGC instance configuration

Each instance (node) of the SS7 SGC Stack exposes information that can be used to check the current instance configuration properties. Current instance configuration properties are queried using `display-local` command of Command-Line Management Console.



See also

[Configuring the SS7 SGC Stack](#) on page 81

15 Alarms

15.1 What are SS7 SGC alarms?

Alarms in the SS7 SGC stack alert the administrator to exceptional conditions. Subsystems in the SS7 SGC stack raise them upon detecting an error condition or an event of high importance. The SS7 SGC stack clears alarms automatically when the error conditions are resolved; an administrator can clear any alarm at any time. When an alarm is raised or cleared, the SS7 SGC stack generates a notification that is sent as a [JMX Notification](#) on page 166 and an [SNMP](#) on page 114 trap/notification.

The SS7 SGC stack defines multiple alarm types. Each alarm type corresponds to a type of error condition or important event (such as "SCTP association down"). The SGC stack can raise multiple alarms of any type (for example, multiple "SCTP association down" alarms, one for each disconnected association).

Alarms are inspected and managed through a set of commands exposed by the Command-Line Management Console, which is distributed with SGC SS7 Stack.



See also

- [Command-Line Management Console](#) on page 173
- [Notifications](#) on page 143 .

Below are details of [Active Alarms and Event History](#) on page 131 , [Generic Alarm Attributes](#) on page 132 , and [Alarm Types](#) on page 133 .

15.2 Active alarms and event history

The SS7 SGC Stack stores and exposes two types of alarm-related information:

- active alarms — a list of alarms currently active
- event history — a list of alarms and notifications that were raised or emitted in the last 24 hours (this is default value — see [Configuring the SS7 SGC Stack](#) on page 81).

At any time, an administrator can clear all or selected alarms.



See also

[Supported CLI Operations for Alarms and Event history](#) on page 184

15.3 Generic alarm attributes

Alarm attributes represent information about events that result in an alarm being raised. Each alarm type has the following generic attributes, plus a group of attributes specific to that alarm type (described in the following sections).

Attribute	Description
<code>id</code>	A unique alarm instance identifier, presented as a number. This identifier can be used to track alarms, for example by using it to identify the raise and clear event entries for an alarm in the event history, or to refer to a specific alarm in the commands which can be used to manipulate alarms.
<code>name</code>	The name of the alarm type. A catalogue on page 133 of alarm types is given below.
<code>severity</code>	alarm severity: <ul style="list-style-type: none">• <code>CRITICAL</code> — application encountered an error which prevents it from continuing (it can no longer provide services)• <code>MAJOR</code> — application encountered an event which significantly impacts delivered services; some services may no longer be available• <code>MINOR</code> — application reports an event which does not have significant impact on delivered services• <code>INFO</code> — application reports an information event which does not have any impact on delivered services• <code>CLEARED</code> — alarm has been cleared.
<code>timestamp</code>	The date and time at which the event occurred.

15.4 Alarm types

This section describes all alarm types that can be raised in an SGC cluster.

15.4.1 General alarms

This section describes the alarms raised concerning the general operational state of the SGC or SGC cluster.

`commswitchbindfailure`

The `commswitchbindfailure` is raised when the [CommSwitch](#) on page is unable to bind to the configured [switch-local-address](#) on page and [switch-port](#) on page for any reason. This is typically caused by misconfiguration; the administrator must ensure that the CommSwitch is configured to use a host and port pair which is always available for the SGC's exclusive use. This alarm is cleared when the CommSwitch is able to successfully bind the configured port.

Attribute	Description	Values of constants
<code>id</code>	unique alarm identifier	
<code>name</code>	name of alarm type	<code>commswitchbindfailure</code>
<code>severity</code>	alarm severity	CRITICAL
<code>timestamp</code>	timestamp when the event occurred	
<code>nodeId</code>	affected node	
<code>failureDescription</code>	the cause of the bind failure	

`distributedDataInconsistency`

The `distributedDataInconsistency` alarm is raised when a distributed data inconsistency is detected. This alarm must be cleared manually since it indicates a problem that may result in undefined behaviour within the SGC, and requires a restart of the SGC cluster to correct. When restarting the cluster it is necessary to fully stop all SGC nodes and only then begin restarting them to properly correct the problem detected by this alarm.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	distributedDataInconsistency
severity	alarm severity	CRITICAL
timestamp	timestamp when the event occurred	
source	the location where the data inconsistency was detected	

nodefailure

The `nodefailure` (node failed) alarm is raised whenever a node configured in cluster is down. It is cleared when an SGC instance acting as that particular node becomes active.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	nodefailure
severity	alarm severity	MAJOR
timestamp	timestamp when the event occurred	
nodeId	affected node	
failureDescription	more information about node failure	

poolCongestion

The `poolCongestion` (Task Pool Congestion) alarm is raised whenever over 80% of a pool's pooled objects are in use. This is typically caused by misconfiguration, see [Static SGC instance configuration](#) on page 81 . It is cleared when less than 50% of pooled objects are in use.



What is a task pool?

A task pool is a pool of objects used during message processing, where each allocated object represents a message that may be processing or waiting to be processed. Each SGC node uses separate task pools for outgoing and incoming messages.

Attribute	Description	Values of constants
<code>id</code>	unique alarm identifier	
<code>name</code>	name of alarm type	<code>poolCongestion</code>
<code>severity</code>	alarm severity	MAJOR
<code>timestamp</code>	timestamp when the event occurred	
<code>poolName</code>	name of the affected task pool	
<code>nodeId</code>	affected node	

poolExhaustion

The `poolExhaustion` (Task Pool Exhaustion) alarm is raised whenever a task allocation request is made on a pool whose objects are all already allocated. This is typically caused by misconfiguration, see [Static SGC instance configuration](#) on page 81 . This alarm must be cleared manually.

Attribute	Description	Values of constants
<code>id</code>	unique alarm identifier	
<code>name</code>	name of alarm type	<code>poolExhaustion</code>

severity	alarm severity	MAJOR
timestamp	timestamp when the event occurred	
poolName	name of the affected task pool	
nodeId	affected node	

workgroupCongestion

The `workgroupCongestion` (Work Group Congestion) alarm is raised when the worker work queue is over 80% occupied. It is cleared when worker work queue is less than 50% occupied.



What is a worker group?

A worker group is a group of workers (threads) that are responsible for processing tasks (incoming/outgoing messages). Each worker has a separate work queue.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	<code>workgroupCongestion</code>
severity	alarm severity	MAJOR
timestamp	timestamp when the event occurred	
nodeID	affected node	
threadIndex	affected worker index	

15.4.2 M3UA alarms

This section describes the alarms raised concerning the M3UA layer of the SGC cluster.

asDown

The `asDown` (Application Server Down) alarm is raised whenever a configured M3UA Application Server is not active. This alarm is typically caused either by a misconfiguration at one or both ends of an M3UA association or by network failure. It is cleared when the Application Server becomes active again.

Attribute	Description	Values of constants
<code>id</code>	unique alarm identifier	
<code>name</code>	name of alarm type	<code>asDown</code>
<code>severity</code>	alarm severity	MAJOR
<code>timestamp</code>	timestamp when the event occurred	
<code>asId</code>	name of affected AS	

asConnDown

The `asConnDown` alarm is raised when an AS connection which was active becomes inactive. This alarm can be caused either by misconfiguration at one or both ends of the M3UA association used, such as by a disagreement on the [routing context to be used](#) on page 137, or by network failure. It is cleared when the Application Server becomes active on the connection.

Attribute	Description	Values of constants
<code>id</code>	unique alarm identifier	
<code>name</code>	name of alarm type	<code>asConnDown</code>
<code>severity</code>	alarm severity	MAJOR

timestamp	timestamp when the event occurred	
asId	name of affected AS	
connectionId	name of the connection on which the affected AS is down	

associationCongested

The `associationCongested` (SCTP association congestion) alarm is raised whenever an SCTP association becomes congested. An association is considered congested if the outbound queue size grows to more than 80% of the configured `out-queue-size` on page 104 for the `connection` on page 104. This alarm is cleared when the outbound queue size drops below 50% of the configured `out-queue-size`.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	<code>associationCongested</code>
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
connectionId	name of affected connection	

associationDown

The `associationDown` (SCTP association down) alarm is raised whenever a configured `connection` on page 104 is not active. This alarm is typically caused either by a misconfiguration at one or both ends of the M3UA association or by network failure. It is cleared when an association becomes active again.

Attribute	Description	Values of constants
id	unique alarm identifier	

name	name of alarm type	associationDown
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
connectionId	name of affected connection	

dpcRestricted

The `dpcRestricted` (destination point code restricted) alarm is raised when the SGC receives a Destination Restricted message from its remote SGP or IPSP peer for a remote destination point code. It is cleared when the DPC restricted state abates on a particular SCTP association.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	dpcRestricted
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
asId	AS on page related to this DPC on page	
dpcId	name fo the affected DPC on page	
connectionId	name of affected connection	

dpcUnavailable

The `dpcUnavailable` (destination point code unavailable) alarm is raised when a configured DPC is unreachable through a particular SCTP association. It is cleared when a DPC becomes reachable again through the particular SCTP association.

Attribute	Description	Values of constants
-----------	-------------	---------------------

id	unique alarm identifier	
name	name of alarm type	dpcUnavailable
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
asId	AS on page related to this DPC on page	
dpcId	name fo the affected DPC on page	
connectionId	name of affected connection	

15.4.3 SCCP alarms

This section describes the alarms raised concerning the SCCP layer of the SGC cluster.

sccpLocalSsnProhibited

The `sccpLocalSsnProhibited` (SCCP local SSN is prohibited) alarm is raised whenever all previously connected TCAP stacks (with the CGIN RA) using a particular SSN become disconnected. This is typically caused either by network failure or administrative action (such as deactivating an RA entity in Rhino). It is cleared whenever at least one TCAP stack using a given SSN connects.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	sccpLocalSsnProhibited
severity	alarm severity	MAJOR
timestamp	timestamp when the event occurred	
ssn	SSN that is prohibited	

sccpRemoteNodeCongestion

The `sccpRemoteNodeCongestion` (SCCP remote node is congested) alarm is raised whenever a remote SCCP node reports congestion. It is cleared when the congestion state abates.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	<code>sccpRemoteNodeCongestion</code>
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
dpc	affected DPC on page	

sccpRemoteNodeNotAvailable

The `sccpRemoteNodeNotAvailable` (SCCP remote node is not available) alarm is raised whenever a remote SCCP node becomes unavailable. It is cleared when the remote node becomes available.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	<code>sccpRemoteNodeNotAvailable</code>
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
dpc	affected DPC on page	

sccpRemoteSsnProhibited

The `sccpRemoteSsnProhibited` (SCCP remote SSN is prohibited) alarm is raised whenever a remote SCCP node reports that a particular SSN is prohibited. It is cleared whenever the remote SCCP node reports that a particular SSN is available.

Attribute	Description	Values of constants
id	unique alarm identifier	
name	name of alarm type	<code>sccpRemoteSsnProhibited</code>
severity	alarm severity	MINOR
timestamp	timestamp when the event occurred	
dpc	affected DPC on page	
ssn	affected SSN on page	

16 Notifications

16.1 What are SS7 SGC notifications?

Notifications in the SS7 SGC stack alert the administrator about infrequent major events. Subsystems in the SS7 SGC stack emit notifications upon detecting an error condition or an event of high importance.

Management clients may use either Java [JMX MBean](#) on page 168 or [SNMP](#) on page 114 trap/notifications to receive notifications emitted by the SS7 SGC stack.

Below are descriptions of:

- [how notifications differ from alarms](#) on page 143
- [generic notification attributes](#) on page 144
- [attributes for specific notification types](#) on page 144 .

You can review the history of emitted notifications using commands exposed by the Command-Line Management Console, which is distributed with the SGC SS7 stack.



See also

[Supported CLI Operations for Alarms and Event History](#) on page

16.2 How are notifications different from alarms?

Notifications are very similar to [Alarms](#) on page 131 in the SGC stack:

- Notifications have attributes (general and notification type-specific attributes).
- The SGC stack stores a history of emitted notifications.



The difference is that notifications are emitted (sent) whenever a particular event occurs; and there is no notion of *active* notifications or a notification being *cleared*.

16.3 Generic notification attributes

Notification attributes represent information about events that result in a notification being emitted. Each notification type has the following generic attributes, plus a group of attributes specific to that notification type (described in the following sections).

Attribute	Description
id	unique notification identifier
name	name of notification type
severity	notification severity: <ul style="list-style-type: none">• <code>CRITICAL</code> — application encountered an error which prevents it from continuing (it can no longer provide services)• <code>MAJOR</code> — application encountered an event which significantly impacts delivered services; some services may no longer be available• <code>MINOR</code> — application reports an event which does not have significant impact on delivered services• <code>INFO</code> — application reports an information event which does not have any impact on delivered services• <code>CLEARED</code> — alarm has been cleared.
timestamp	timestamp when the event occurred

16.4 Notification types

This section describes all notification types that can be emitted by the SGC node:

- [MTP decode errors](#) on page 145
- [SCCP decode errors](#) on page 146
- [TCAP decode errors](#) on page 146
- [TCAP stack register](#) on page 147
- [TCAP stack unregister](#) on page 148
- [MTP decode errors](#) on page 145

16.4.1 MTP decode errors

Notification name `mtpDecodeErrors` (observed errors in incoming messages at the MTP layer) contains information about the number of badly formatted messages at the MTP layer and the number of messages directed to an unsupported MTP user. This notification is emitted periodically, with a summary of the number of errors in that period; it is not emitted if there are no errors.

Attribute	Description	Values of constants
<code>id</code>	unique notification identifier	
<code>name</code>	name of notification type	<code>mtpDecodeErrors</code>
<code>severity</code>	notification severity	INFO
<code>timestamp</code>	timestamp when the event occurred	
<code>nodeId</code>	affected node	
<code>connectionID</code>	affected connection	
<code>decodeFailuresLastTime</code>	number of message decode failures during report interval	
<code>unsupportedUserFailuresTime</code>	number of messages with unsupported user part during report interval	

16.4.2 SCCP decode errors

Notification name `sccpDecodeErrors` (observed errors in incoming messages at the SCCP layer) contains information about the number of badly formatted messages at the SCCP layer and the number of messages directed to a prohibited SSN. This notification is emitted periodically, with a summary of the number of errors in that period; it is not emitted if there are no errors.

Attribute	Description	Values of constants
<code>id</code>	unique notification identifier	
<code>name</code>	name of notification type	<code>sccpDecodeErrors</code>
<code>severity</code>	notification severity	INFO
<code>timestamp</code>	timestamp when the event occurred	
<code>nodeId</code>	affected node	
<code>decode-failures-last-time</code>	number of message decode failures during report interval	
<code>ssn-failures-last-time</code>	number of messages directed to prohibited SSN during report interval	

16.4.3 TCAP decode errors

Notification name `tcapDecodeErrors` (observed errors in incoming messages at TCAPthe layer) contains information about number of badly formatted messages at the TCAP layer and number of messages that SGC is unable to forward to TCAP Stack (CGIN RA). This notification is emitted periodically, with a summary of the number of errors in that period; it is not emitted if there are no errors.

Attribute	Description	Values of constants
<code>id</code>	unique notification identifier	

name	name of notification type	tcapDecodeErrors
severity	notification severity	INFO
timestamp	timestamp when the event occurred	
nodeId	affected node	
decode-failures-last-time	number of message decode failures during report interval	
missing-peer-last-time	number of messages that SGC was unable to forward to the TCAP stack	

16.4.4 TCAP stack register

Notification name `tcapStackRegister` (observed errors in incoming messages at the TCAP layer) is emitted whenever a TCAP stack registers in the SGC.

Attribute	Description	Values of constants
id	unique notification identifier	
name	name of notification type	tcapStackRegister
severity	notification severity	INFO
timestamp	timestamp when the event occurred	
ssn	registered SSN	
nodeId	affected node	
stackIP	IP address of the stack	

prefix	allocated transaction prefix	
--------	------------------------------	--

16.4.5 TCAP stack unregister

Notification name `tcapStackUnregister` (observed errors in incoming messages at the TCAP layer) is emitted whenever a TCAP Stack deregisters in the SGC.

Attribute	Description	Values of constants
id	unique notification identifier	
name	name of notification type	<code>tcapStackUnregister</code>
severity	notification severity	INFO
timestamp	timestamp when the event occurred	
ssn	registered SSN	
node-id	affected node	
stack-ip	IP address of the stack	
prefix	allocated transaction prefix	

17 Statistics

17.1 What are SS7 SGC statistics?

Statistics are usage, status, and health information exposed by the [SS7 SGC](#) on page [173](#) stack. Each layer of the SS7 SGC stack exposes a set of statistics that provide insight into the current operational state of the cluster. The statistics data is available cluster-wide, independent of which particular node happens to be responsible for gathering a particular set of statistics.



Statistics update interval

To maximize performance, statistics exposed by SGC cluster nodes are updated periodically (every 1 second). So any management tools pooling SGC statistics should use a pooling interval that is $> 1s$.

The SGC statistics subsystem collects multiple types of statistical data, which can be separated in two broad categories:

- frequently updated statistic data types:
 - counter — monotonically increasing integer values; for example the number of received messages
 - gauge — the amount of a particular object or item, which can increase and decrease within some arbitrary bound, typically between 0 and some positive number; for example, the number of processing threads in a worker group.
- character values that can change infrequently or not at all; these include:
 - identifying information — usually a name of an object for which a set of statistics is gathered; for example, the name of a node
 - status information — information about the state of a given configuration object; for example, the SCTP association state `ESTABLISHED`.

The SGC statistics subsystem is queried using commands exposed by the [Command-Line Management Console](#) on page 173 , which is distributed with SGC SS7 Stack.

17.2 Statistics

Below are details of the exposed subsystem-specific statistics:

- [M3UA](#) on page 150
 - [AsInfo](#) on page 150
 - [AssociationInfo](#) on page 151
 - [DpcInfo](#) on page 152
- [SCCP](#) on page 153
 - [LocalSsnInfo](#) on page 153
 - [OgtInfo](#) on page 154
 - [PcInfo](#) on page 155
 - [RemoteSsnInfo](#) on page 156
- [TCAP](#) on page 156
 - [TcapConnInfo](#) on page 156
- [TOP](#) on page 157
 - [HealthInfo](#) on page 157

17.2.1 M3UA

The M3UA layer exposes statistical data about [AS](#) on page 150 , [Association](#) on page 151 , and [DPC](#) on page 152 .



The JMX Object name for M3UA statistics is `SGC:module=m3ua,type=info`

AsInfo

The set of configured Application Servers currently associated with connections.

Statistic	Description
-----------	-------------

RXCount	number of received messages directed to this AS through a particular connection
TXCount	number of sent messages originating from this AS through a particular connection
asId	name of the AS
connectionId	name of the connection
nodeId	name of the node
status	status of this AS as seen through a particular connection (possible values: INACTIVE , ACTIVE)



This information is also exposed through multiple MBeans named `SGC:type=info,module=m3ua,elem=AsInfo,id=CONNECTION_NAME&AS_NAME` , where `id` is a concatenation of the relevant connection and AS names.

AssociationInfo

The set of SCTP associations defined within the cluster.

Statistic	Description
RXDataCount	number of received M3UA Payload Data Messages
RXErrorCount	number of received M3UA Management Error Messages
RXSctpCount	number of received M3UA Messages
TXDataCount	number of transmitted M3UA Payload Data Messages
TXErrorCount	number of transmitted M3UA Management Error Messages
TXSctpCount	number of transmitted M3UA Messages

connectionId	name of connection
nodeId	name of node
numberOfInStreams	number of input streams
numberOfOutStreams	number of output streams
outQSize	number of messages waiting to be written into SCTP association
peerAddresses	peer IP addresses
sctpOptions	socket options associated with an SCTP channel
status	status of this connection (possible values: INACTIVE , ESTABLISHED)



This information is also exposed through multiple MBeans named `SGC:type=info,module=m3ua,associationInfo,id=CONNECTION_NAME` , where `id` is the name of the relevant connection.

DpcInfo

The set of Destination Point Codes that are associated with connections at this point in time.

Statistic	Description
ConnectionId	name of connection
Dpc	destination point code
DpcId	name of DPC
Status	status of this DPC (possible values: ACTIVE , CONGESTED , INACTIVE , RESTRICTED)



This information is also exposed through multiple MBeans named `SGC:type=info,module=m3ua,dpcInfo,id=CONNECTION_NAME&DPC`, where `id` is a concatenation of the relevant connection name and DPC.

17.2.2 SCCP

The SCCP layer exposes statistic data about the [local SSN](#) on page 153, [GT translation rules \(outgoing\)](#) on page 154, [PC routing](#) on page 155, and [remote SSN](#) on page 156.



The JMX Object name for SCCP statistics is `SGC:module=sccp,type=info`

LocalSsnInfo

The set of local SSN; that is, TCAP Stacks (CGIN RAs, scenario simulators) connected to the cluster representing particular SSNs.

Statistic	Description
RXCount	number of messages received by the SSN
TXCount	number of messages originated by the SSN
ssn	local SSN
status	status of this SSN (possible values: PROHIBITED, CONGESTED, ALLOWED)
NUnitdataReqCount	number of N-UNITDATA req primitives received by the SSN
udtSentCount	number of UDT messages sent by this SSN
xudtSentNoSegmentationCount	number of XUDT messages sent by this SSN without a segmentation parameter
xudtSentSegmentationCount	number of XUDT messages sent by this SSN with a segmentation parameter

segmentationFailureCount	number of segmentation failures at this SSN
udtReceivedCount	number of UDT messages received by this SSN
xudtReceivedNoSegmentationCount	number of XUDT messages received by this SSN without a segmentation parameter
xudtReceivedSegmentationCount	number of XUDT messages received by this SSN with a segmentation parameter
NUnitdataIndCount	number of N-UNITDATA ind primitives received by this SSN
reassembledNUnitdataIndCount	number of reassembled N-UNITDATA ind primitives received by this SSN
reassemblyFailureCount	number of reassembly failures at this SSN



This information is also exposed through multiple MBeans named `SGC:type=info,module=sccp,elem=LocalSsnInfo,ssn=SSN_NUMBER`, where `ssn` is the value of the relevant local SSN.

OgtInfo

The set of Global Title translation rules currently used for outgoing messages. Different translation rules might be in effect at the same time on different nodes, depending on the `prefer-local` attribute of the `outgoing-gtt` configuration object.

The set of currently used translation rules depends on:

- the priority attribute of the `outgoing-gtt` configuration object
- the priority attribute of the `route` configuration object
- DPC reachability status (GT translations that result in unreachable DPC are ignored).



Local cluster PC

Entries with an empty `connId` attribute and an `rc` attribute value of `-1` represent GT translations that result in a PC of the SGC Cluster itself. Messages that are to be routed to a local PC are load balanced among TCAP Stacks within the local cluster.



See also

- [M3UA Configuration](#) on page 97
- [SCCP Configuration](#) on page 107

Statistic	Description
<code>addrInfo</code>	address string
<code>addrType</code>	values of attributes <code>trtype</code> , <code>numplan</code> , <code>natofaddr</code> , and <code>isPrefix</code> , as defined for this rule
<code>connId</code>	name of connection on which messages with matching GT will be transmitted
<code>dpc</code>	destination point code to which messages with matching GT will be transmitted
<code>nodeId</code>	name of the node on which translation rule is used
<code>rc</code>	routing context which will be used when transmitting messages with matching GT
<code>routeNodeId</code>	name of node where SCTP connection is established

PcInfo

The set of associations and RCs across which traffic for a given DPC are currently load balanced by each node.



Local cluster PC

Entries with an empty `connId` attribute and an `rc` attribute value of `-1` represent nodes of the SGC Cluster itself. Messages that are to be routed to the local DPC are load balanced among TCAP Stacks within the local cluster.

Statistic	Description
-----------	-------------

connId	name of connection on which messages with matching DPC will be transmitted
dpc	destination point code
nodeId	name of node where connection is established
rc	routing context which will be used when transmitting messages with matching DPC

RemoteSsnInfo

The set of remote SSNs for which their status is known.

Statistic	Description
dpc	destination point code
ssn	remote SSN
status	status of this SSN (possible values: PROHIBITED , CONGESTED , ALLOWED)

17.2.3 TCAP

The TCAP layer exposes statistics data about [connected TCAP Stacks](#) on page 156 .



The JMX Object name for TCAP statistics is `SGC:module=tcap,type=info`

TcapConnInfo

The set of connected TCAP stacks (such as the CGIN RA).

Statistic	Description
QSize	number of messages waiting to be written into the TCP connection

RXCount	number of messages directed from SGC to TCAP stack
TXCount	number of messages directed from TCAP stack to SGC
connectTime	time and date when connection was established
localIp	local SGC IP used by connection
localPort	local SGC port used by connection
prefix	transaction prefix assigned to TCAP stack
remoteIp	IP from which TCAP stack originated connection to SGC
remotePort	port from which TCAP stack originated connection to SGC
serviceNodeId	node name to which TCAP stack is connected
ssn	SSN that TCAP stack is representing



This information is also exposed through multiple MBeans named `SGC:type=info,module=tcap,elem=TcapConnInfo,ssn=SSN_NUMBER,prefix=ASSIGNED_PREFIX`, where `ssn` is the SSN represented by the connected TCAP stack, and `prefix` is the transaction-ID prefix assigned to the connected TCAP stack.

17.2.4 TOP

The SGC node processing-load statistics information.



The JMX Object name for TOP statistics is `SGC:module=top,type=info`

HealthInfo

The set of current load-processing statistics for nodes in the cluster.

Statistic	Description
allocatedIndTasks	number of allocated task objects, each of which represents an incoming message that may be processing or waiting to be processed
allocatedReqTasks	number of allocated task objects, each of which represents an outgoing message that may be processing or waiting to be processed
contextExecutionCount	cumulative number of tasks representing incoming or outgoing messages that have finished processing
contextExecutionTime	cumulative wall-clock time in milliseconds that tasks representing incoming or outgoing messages were in the allocated state
executorQueueSize	number of background tasks mainly related to the SCCP management state, queued for execution nodeId name of node
workerExecutionCount	cumulative number of tasks (for incoming or outgoing messages) processed by worker threads
workerExecutionTime	cumulative wall-clock time in milliseconds spent by worker threads processing tasks (for incoming or outgoing messages)
workerGroupSize	number of worker threads used to process tasks (for incoming or outgoing messages)
forceAllocatedIndTasks	cumulative number of task objects that had to be force allocated (i.e. not from the task pool) in order to process incoming messages
forceAllocatedReqTasks	cumulative number of task objects that had to be forced allocated (i.e. not from the task pool) in order to process outgoing messages

18 Logging

18.1 About the Logging subsystem

The Logging subsystem in the SGC Stack is based on the Simple Logging Facade for Java (SLF4J), which serves as a simple facade or abstraction for various logging frameworks, such as `java.util.logging`, `logback`, and `log4j`. SLF4J allows the end user to plug in the desired logging framework at -deployment- time. That said, the standard SGC Stack distribution uses SLF4J backed with the [Apache Log4J logging architecture](#) (version 1.x).

By default, the SGC stack outputs a minimal set of information, which will be used before the logging subsystem has been properly initialized, or when a severe error includes a logging subsystem malfunction. The default start-up script for the SGC stack redirects the standard output to `SGC_HOME/logs/startup.<timestamp>`, where `<timestamp>` denotes the SGC start time. This file is rolled over after reaching a size of 100 MB, with at most 10 backups.



SGC_HOME SGC_HOME here represents the path to the SGC Stack installation directory.

18.2 Logger names, levels, appenders

Log4J logging architecture includes [logger names](#) on page 159 , [log levels](#) on page 160 , and [log appenders](#) on page 160 .

18.2.1 Logger names

Subsystems within the SGC stack send log messages to specific **loggers** . For example, the `alarming.log` logger receives messages about alarms being raised.

Examples of logger names include:

- `root` — the root logger, from which all loggers are derived (can be used to change the log level for all loggers at once)
- `com.cts.ss7.SGCShutdownHook` — for log messages related to the SGC shutdown process
- `com.cts.ss7.sccp.layer.scmg.SccpManagement` — for log messages related to incoming SCCP management messages.

18.2.2 Log levels

Log levels can be assigned to individual loggers to filter how much information the SGC produces:

Log level	Information sent
OFF	no messages sent to logs (<i>not recommended</i>)
FATAL	error messages for unrecoverable errors only (<i>not recommended</i>)
ERROR	error messages (<i>not recommended</i>)
WARN	warning messages
INFO	informational messages (the default)
DEBUG	messages containing useful debugging information
TRACE	messages containing verbose debugging information

Each log level will log all messages for that log level and above. For example, if a logger is set to the `WARN` level, all of the log messages logged at the `WARN` , `ERROR` , and `FATAL` levels will be logged as well.

If a logger is not assigned a log level, it inherits its parent's. For example, if the `com.cts.ss7.SGCShutdownHook` logger has not been assigned a log level, it will have the same effective log level as the `com.cts.ss7` logger.

The `root` logger is a special logger which is considered the parent of all other loggers. By default, the `root` logger is configured with the `DEBUG` log level, but the `com` logger parent of all loggers used by SGC overwrites it to the `WARN` log level. In this way, all other SGC loggers will output log messages at the `WARN` log level or above unless explicitly configured otherwise.

18.2.3 Log appenders

Log appenders append log messages to destinations such as the console, a log file, socket, or Unix syslog daemon. At runtime, when SGC logs a message (as permitted by the log level of the associated logger), SGC sends the message to the log appender for writing. Types of log appenders include:

- **file appenders** — which append messages to files (and may be [rolling file appenders](#) on page 161)
- **console appenders** — which send messages to the standard out
- **custom appenders** — which you configure to receive only messages for particular loggers.

Rolling file appenders

Typically, to manage disk usage, administrators are interested in sending log messages to a set of rolling files. They do this by setting up rolling file appenders which:

- create new log files based on file size / daily / hourly
- rename old log files as numbered backups
- delete old logs when a certain number of them have been archived



You can configure the size and number of rolled-over log files.

18.3 Default logging configuration

The default SGC Stack logging configuration is an XML file, in a format supported by Log4J, loaded during SGC startup. The default configuration is stored in `SGC_HOME/config/log4j.xml` . The related DTD file is stored in `SGC_HOME/config/log4j.dtd` .




For more information on Log4J and Log4J XML-based configuration, please see:

- [Log4J version 1.2 manual](#)
- [Log4j XML Configuration Primer](#)

18.3.1 Default appenders

The SGC Stack comes configured with the following appenders:

Appender	Where it sends messages	Logger name	Type of appender
----------	-------------------------	-------------	------------------

fileAppender	the SGC logs directory (<code>SGC_HOME/logs/ss7.log</code>); file is rolled over every hour; number of rolled over log files is not limited (by default, no log files are removed)	root	a rolling file appender
stdoutAppender	<p>the SGC console where a node is running (to standard output stream)</p> <div><p>By default, the value of the threshold attribute is set to <code>OFF</code> (which effectively disables logging to the console).</p></div>	root	a console appender

19 JMX MBean Naming Structure



MBeans exposing operational state

SS7 SGC Stack operational state is exposed through a set of JMX MBeans. Description of these MBeans and their naming conventions might be of interest to a user wishing to create a custom management tool for the SGC stack.

19.1 Instance Management MBean

Each instance (node) of the SS7 SGC Stack exposes information that can be used to check the current instance configuration properties.



The JMX Object name for the MBean exposing this information is `SGC:type=local`.

This bean exposes following [attribute](#) on page 163 and [operations](#) on page 164.


19.1.1 Properties attribute

The properties attribute is a collection of records. Each such record represents runtime properties of the currently connected SGC Stack instance (that is, the instance to which the JMX management client is connected). For more about instance configuration, please see [Configuring the SS7 SGC Stack](#) on page 81. Below are properties record fields:

Name	Description
key	property key
description	property description
defaultValue	value used if <code>configuredValue</code> is not provided

19.1.2 Operations

The results of JMX MBean operations invoked on the instance-management bean are local to the currently connected SGC Stack instance (that is, the instance to which the JMX management client is connected).

Name	Description
checkAlive	tests if the JMX RMI connector of the currently connected SGC Stack responds to the invocation of a JMX MBean operation
shutdown	<p>attempts to shutdown the currently connected SGC Stack instance</p> <div>  <p>When this operation is successful, it will result in the JMX management client being disconnected and reporting an error, as the SGC stack instance was shutdown.</p> </div>

19.2 Alarm MBean naming structure

Every type of alarm used by the SS7 SGC stack is represented by a separate MBean. The names of MBeans for SGC alarms use the common domain SGC and this set of properties: `type` , `subtype` , `name` . The values of `type` and `subtype` are the same for all alarm MBeans; the value of the `name` property represents a type of alarm. Whenever an actual alarm of a given type is raised, a new MBean representing that instance of the alarm is created. The name of that MBean has an additional property, `id` , which uniquely identifies that alarm instance in the SGC cluster.

For example, an alarm type representing information about an SCTP association (connection) going down has the MBean name `SGC:type=alarming,subtype=alarm,name=associationDown` . Whenever a particular association is down, the MBean representing the error condition for that particular association is created with a name such as `SGC:type=alarming,subtype=alarm,name=associationDown,id=302` , where the `id` property value is unique among all alarms in the cluster.

Most GUI-based JMX Management tools represent the naming space as a tree of MBean objects, like this:

MBeans

- ▶ JMImplementation
- ▼ SGC
 - ▼ alarming
 - ▼ alarm
 - ▶ asDown
 - associationCongested
 - ▼ associationDown
 - 302
 - dpcRestricted
 - dpcUnavailable
 - ipspAsDown
 - nodefailure
 - poolCongestion
 - sccpLocalSsnProhibited
 - sccpRemoteNodeCongestion
 - ▼ sccpRemoteNodeNotAvailable
 - 299
 - sccpRemoteSsnProhibited
 - workgroupCongestion
 - alarm-history
 - alarm-table

Attributes | Operations | Notifications | Me

Attribute values

Name	Value
connectionId	N1-LE1-CSG
id	302
name	associationDown
severity	MINOR
timestamp	Thu Jan 23 11:11:10 CET 2014

19.3 Alarm raised and cleared notifications

Whenever SGC raises or clears an alarm, a JMX notification is emitted. To be notified of such events, each MBean representing an alarm type supports `alarm.onset` and `alarm.clear` notifications. An administrator can use a JMX management tool and subscribe to such events for each alarm type.

Here's an example view of a JMX Management tool reporting received notifications:

The screenshot shows a JMX Management tool interface with a tree view on the left and a notification buffer on the right.

MBeans Tree:

- JMImplementation
 - SGC
 - alariming
 - alarm
 - asDown
 - associationCongested
 - associationDown
 - dpcRestricted
 - dpcUnavailable
 - ipspAsDown
 - nodeFailure
 - poolCongestion
 - sccpLocalSsnProhibited** (highlighted)
 - sccpRemoteNodeCongestic
 - sccpRemoteNodeNotAvailat
 - sccpRemoteSsnProhibited
 - workgroupCongestion
 - alarm-history

Notification Buffer:

TimeStamp	Type	UserData	SeqNum	Message	Event	Source
17:24:25:840	alarm.clear	ssn=101 id=299 timestamp= severity=Cl name=sccpLo	299	Alarm clear	javax.manage...	SGC:type=alarmin...

Buttons at the bottom: [Subscribe](#) [Unsubscribe](#) [Clear](#)

19.4 Active alarms and alarm history

Besides alarm type-specific MBeans, the SS7 SGC stack exposes two generic MBeans that enable review of active alarms and history of alarms that were raised and subsequently cleared during cluster operation:

- SGC:type=alariming,subtype= [alarm-table](#) on page 167 — currently active alarms
- SGC:type=alariming,subtype= [alarm-history](#) on page 168 — history of alarms

19.4.1 Alarm Table

The Alarm Table MBean contains a single attribute, `EventList`, which is a list of records representing attributes of currently active alarms. Alarm type-specific attributes are represented within a single parameters column.

Here's an example view of Alarm Table MBean attributes in a JMX management tool:

Attributes | Operations | Notifications | Metadata

Attribute values

Name	Value														
EntryList	< Tabular Navigation >														
	<< < Composite Navigation 1/3 >														
	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>description</td><td>AS identified by asId is DOWN</td></tr><tr><td>id</td><td>296</td></tr><tr><td>name</td><td>asDown</td></tr><tr><td>parameters</td><td>asId='ASP-RC1'</td></tr><tr><td>severity</td><td>MAJOR</td></tr><tr><td>timestamp</td><td>Thu Jan 23 11:11:08 CET 2014</td></tr></tbody></table>	Name	Value	description	AS identified by asId is DOWN	id	296	name	asDown	parameters	asId='ASP-RC1'	severity	MAJOR	timestamp	Thu Jan 23 11:11:08 CET 2014
	Name	Value													
	description	AS identified by asId is DOWN													
	id	296													
	name	asDown													
	parameters	asId='ASP-RC1'													
severity	MAJOR														
timestamp	Thu Jan 23 11:11:08 CET 2014														

The Alarm Table MBean exposes a `clearAlarm` operation that can be used to clear any raised alarm based on its `id`.

19.4.2 Alarm History

The Alarm History MBean contains a single attribute, `AlarmHistory`, which is a list of records representing attributes of raised and optionally cleared alarms within the last 24 hours (This period is customizable, as described in [Configuring the SS7 SGC Stack](#) on page 81). Alarm type-specific attributes are represented within a single parameters column. For each alarm that was raised and subsequently cleared, there are two records available in the list: one is generated when the alarm is raised (with a severity value other than `CLEARED`); the second record is generated when the alarm is cleared (with a severity value of `CLEARED`).

Information concerning [notifications](#) on page 143 is also recorded in the Alarm History MBean.

The Alarm History MBean exposes a clear operation that can be used to clear all entries stored in the alarm history.

19.5 Notification MBean naming structure

Every type of notification used by the SS7 SGC stack is emitted by a separate MBean. SGC-alarm MBean names use the common domain `SGC`, and this set of properties: `type`, `subtype`, `name`. Values of `type` and `subtype` are the same for all notification MBeans; the value of the `name` property represents a type of notification.

For example, the MBean type emitting notifications about incoming SCCP-message decoding errors is represented by this MBean name:

`SGC:type=alarming,subtype=notif,name=sccpDecodeErrors`. Whenever there is an error during decoding of an SCCP message, a JMX notification is emitted by this MBean. (Actually, for this specific notification, a single notification summarizes multiple errors over a specific time interval.)

Most GUI-based JMX Management tools represent the notification naming space as a tree of MBean objects, like this:

The screenshot displays the JMX console interface. On the left, the MBean tree is expanded to show the 'com.sun.management' package, with 'tcapStackUnregister' selected. On the right, the 'Notifications[1]' tab is active, showing a 'Notification buffer' table. The table has columns for 'TimeStamp', 'Type', 'UserData', 'SeqNum', 'Message', 'Event', and 'Source'. A single notification is listed with the following values: TimeStamp: 17:47:40:139, Type: info, UserData: id=302, timestamp=..., name=tcapS..., prefix=419..., ssn=101, SeqNum: 302, Message: Stack has unre..., Event: javax.manage..., and Source: SGC:type=alarmin... Below the table are buttons for 'Subscribe', 'Unsubscribe', and 'Clear'.

19.6 Statistics MBean naming structure

SGC statistics are exposed through a set of JMX Management MBeans. This section describes the MBean naming structure of subsystem statistics and processing object-specific statistics.

19.6.1 Subsystem statistics

SGC stack subsystems expose multiple JMX MBeans containing statistical information. SGC statistics MBeans names use the common domain SGC and this set of properties: `module`, `type`. The value of `type` is the same for all statistics MBeans; the value of the `module` property represents a subsystem. Attributes of these beans are arrays of records; each record contains statistical information related to a processing object configured within the SGC cluster (such as a particular SCTP association).

For example, the statistics MBean representing information about the M3UA subsystem (layer) is represented by the MBean named `SGC:module=m3ua,type=info`. This MBean has (among others) the attribute `DpcInfo`, which contains information about DPC (destination point code) reachability through a particular connection; for example:

Attribute	Value
-----------	-------

connectionId	N1-LE1-CSG
dpc	4114
dpcId	2-2-2
status	DOWN

Most GUI-based JMX Management tools represent the naming space as a tree of MBean objects, like this:

The screenshot shows a JMX Management tool interface. On the left is a tree of MBean objects under the 'MBeans' root. The tree structure is as follows:

- MBeans
 - JMImplementation
 - SGC
 - alarming
 - general
 - info
 - m3ua (selected)
 - sccp
 - tcap
 - top
 - m3ua
 - sccp
 - snmp
 - com.sun.management
 - java.lang
 - java.nio
 - java.util.logging

On the right, the 'Attributes' tab is selected, showing the 'Attribute values' for the 'DpcInfo' MBean. The values are as follows:

Name	Value
AsInfo	javax.management.openmbean.CompositeData[1]
AssociationInfo	javax.management.openmbean.CompositeData[1]
< Tabular Navigation >	
<< < Composite Navigation > >>	
Name	Value
connectionId	N1-LE1-CSG
dpc	4114
dpcId	2-2-2
status	DOWN

19.6.2 Processing object-specific statistics

The SGC stack also exposes processing object-specific MBeans, such as an MBean containing statistics created for each connection. Information exposed through such MBeans is exactly equivalent to that exposed through subsystem statistic MBeans. SGC processing object-specific MBean names use the common domain `SGC` and this set of properties: `module`, `type`, `element`, `id`.

- The value of `type` is the same for all statistics MBeans.
- The value of `module` represents a subsystem.
- The value of `elem` identifies a group of processing objects of the same type (such as connections or nodes).
- The value of `id` identifies a particular processing object (such as a particular connection or particular node).

For example, the statistics MBean representing information about network reachability for a particular DPC through a particular connection in the M3UA subsystem (layer) is represented the MBean named `SGC:type=info,module=m3ua,elem=dpcInfo,id=N1-LE1-CSG&4114`. Information exposed through this MBean is exactly the same in the example above:

Attribute	Value
<code>connectionId</code>	<code>N1-LE1-CSG</code>
<code>dpc</code>	<code>4114</code>
<code>dpcId</code>	<code>2-2-2</code>
<code>status</code>	<code>DOWN</code>

Most GUI-based JMX Management tools represent the naming space as a tree of MBean objects, like this:

MBeans

- ▶ JMImplementation
- ▼ SGC
 - ▶ alarming
 - ▶ general
 - ▼ info
 - ▼ m3ua
 - ▶ AsInfo
 - ▶ associationInfo
 - ▼ dpcInfo
 - N1-LE1-CSG&4114**
 - ▼ sccp
 - ▶ LocalSsnInfo
 - tcap
 - top

Attributes | Operations | Notifications | Metadata

Attribute values

Name	Value
ConnectionId	N1-LE1-CSG
Dpc	4114
DpcId	2-2-2
Status	DOWN

20 Command-Line Management Console

20.1 What is the SGC Stack Command-Line Console?

The SGC Stack Command-Line Management Console is a [CLI](#) on page tool working in interactive and non-interactive mode to manage and configure the SGC Stack cluster. The Command-Line Console uses JMX Beans exposed by the SGC Stack to manage cluster configuration. The command syntax is based on ITU-T [MML](#) on page Recommendations Z.315 and Z.316.

20.2 Installation and requirements

Command-Line Console installation requires unpacking the tar.gz archive file in any location. Unpacked, the folder structure is:

File/Directory	Description
.	SGC CLI installation directory
./conf	logging configuration and CLI settings file
./lib	Java libraries used by SGC CLI.
./log	log file
./sgc-cli.sh	CLI start-up script



JAVA_HOME

The SGC CLI start-up script expects the `JAVA_HOME` environment variable to be set up and point to a valid Java Virtual Machine version 7 or greater (it expects executable file `JAVA_HOME/bin/java`).

20.3 Working with the SGC CLI

The SGC CLI should be started by executing the `sgc-cli.sh` script. Default connection settings point to the SGC JMX Server exposed at:

```
host: 127.0.0.1
port: 10111
```

Here is an example of starting the CLI with the default IP and port:

```
./sgc-cli.sh
```

Here is an example of starting the CLI with an alternative IP and port setup:

```
./sgc-cli.sh -h 192.168.1.100 -p 10700
```

The SGC CLI supports other configuration parameters, which you can display by executing the startup script with the `-?` or `--help` options:

```
usage: sgc-cli.sh [-?] [-b <FILE>] [-h <HOST>] [-P <PASSWORD>] [-p
<PORT>] [-stopOnError <true/false>] [-U <USER>] [-x <FILE>]
+-----Options list-----+
-?,--help Displays usage
-b,--batch <FILE> Batch file
-h,--host <HOST> JMX server user
-P,--pass <PASSWORD> JMX password
-p,--port <PORT> JMX server port
-ssl,--ssl <true/false> JMX connection SSL enabled
-stopOnError,--stopOnError <true/false> Stop when any error occurs in
batch command
-U,--user <USER> JMX user
-x,--export <FILE> File where the configuration
dump will be saved
+-----+

```

20.3.1 Enabling secured JMX server connection

CLI supports secured JMX server connection through the [SSL](#) on page protocol (SSL). It can be enabled in two ways:

- by specifying the `ssl=true` property in `conf/cli.properties`,
- by adding the configuration parameter `-ssl true` to the start script.



The configuration parameter value takes precedence over the value defined in `conf/cli.properties`.

The SSL connection (in both cases) also requires specifying additional properties in the `conf/cli.properties` file, for the `trustStore`, `keyStore` password and location. Below is a sample configuration:

```
#####
###SSL configuration###
#####
#File location relative to conf folder
javax.net.ssl.keyStore=sgc-client.keystore
javax.net.ssl.keyStorePassword=changeit
#File location relative to conf folder
javax.net.ssl.trustStore=sgc-client.keystore
javax.net.ssl.trustStorePassword=changeit
```

20.3.2 Basic command format

The command syntax is based on ITU-T [MML](#) on page Recommendations Z.315 and Z.316:

command: `[parameter-name1=parameter-value1][,parameter-name2=value2]...;`

Where:

- `command` is the name of the command to be executed
- `parameter-name` is the name of command parameter
- `parameter-value` is the value of the command parameter.



Command parameters are separated by commas (,).

When specifying a command with no parameters, the colon (:) is optional.

The ending semicolon (;) is optional.

20.3.3 MML syntax auto-completing

The CLI supports MML syntax auto-completing on the command name and command parameters level. Pressing the **<TAB>** key after the prompt will display all available command names:

```
SGC[127.0.0.1:10111]><TAB_PRESSED>
batch clear-active-alarm clear-all-alarms
create-as create-as-connection create-as-precond
create-conn-ip create-connection create-cpc
create-dpc create-inbound-gtt create-local-endpoint
create-local-endpoint-ip create-node create-outbound-gt
create-outbound-gtt create-replace-gt create-route
create-snmp-node create-target-address create-usm-user
disable-as disable-connection disable-local-endpoint
disable-node disable-snmp-node display-active-alarm
display-as display-as-connection display-as-precond
display-conn-ip display-connection display-cpc
display-dpc display-event-history display-inbound-gtt
display-info-asinfo display-info-associationinfo display-info-dpcinfo
display-info-healthinfo display-info-localssninfo display-info-ogtinfo
display-info-opcinfo display-info-remotessninfo display-info-tcapconninfo
display-local-endpoint display-local-endpoint-ip display-node
display-outbound-gt display-outbound-gtt display-parameters
display-replace-gt display-route display-snmp-node
display-target-address display-usm-user enable-as
enable-connection enable-local-endpoint enable-node
enable-snmp-node exit export
help modify-as modify-as-connection
modify-connection modify-dpc modify-inbound-gtt
modify-local-endpoint modify-node modify-outbound-gt
```



```

modify-outbound-gtt modify-parameters modify-replace-gt
modify-snmp-node modify-target-address modify-usm-user
remove-as remove-as-connection remove-as-precond
remove-conn-ip remove-connection remove-cpc
remove-dpc remove-inbound-gtt remove-local-endpoint
remove-local-endpoint-ip remove-node remove-outbound-gt
remove-outbound-gtt remove-replace-gt remove-route
remove-snmp-node remove-target-address remove-usm-user
SGC[127.0.0.1:10111]>

```

When you press the **<TAB>** key after providing a command name, the CLI displays all available parameters for that command.

Pressing **<TAB>** after providing command parameters displays legal values (for enumeration, reference, and boolean parameters). For example:

```

SGC[127.0.0.1:10111]> modify-connection: <TAB_PRESSED>
oname=A-CONN-1 oname=B-CONN-1
SGC[127.0.0.1:10111]> modify-connection: oname=

```

20.3.4 Help mode

You can access the SGC CLI's built in help system by either:

- executing the command `help: topic=topicName`
- switching to help mode, by executing the manual `help` command (with no parameters).

Help mode displays topics that you can access. (Alternatively, you can press the **<TAB>** to display available values of a topic command parameter. The list of available topics in manual help mode looks like this:

```

SGC[127.0.0.1:10111]> help
Executing help manual...
Use <TAB> to show topic list. Write 'topic name' to see its description. Use exit command if you want to
quit the manual.
Hint: 'create-, display-, remove-, modify-, enable-, disable-' operations are described by single topic
for given MBean name.

```

Available topics:

```
as as-connection as-precond batch clear-active-alarm clear-all-alarms
conn-ip connection cpc display-active-alarm display-event-history display-info-asinfo
display-info-associationinfo display-info-dpcinfo display-info-healthinfo display-info-localssninfo
display-info-ogtinfo display-info-opcinfo
display-info-remotessninfo display-info-tcapconninfo dpc exit export help
inbound-gtt local-endpoint local-endpoint-ip node outbound-gt outbound-gtt
parameters replace-gt route snmp-node target-address usm-user
help>
```

The result of executing a help topic command looks like this:

```
SGC[127.0.0.1:10111]> help: topic=<TAB_PRESSED>
topic=as topic=as-connection topic=as-precond topic=batch topic=clear-active-alarm
topic=clear-all-alarms topic=conn-ip topic=connection topic=cpc topic=display-active-alarm
topic=display-event-history topic=display-info-asinfo topic=display-info-associationinfo topic=display-
info-dpcinfo topic=display-info-healthinfo
topic=display-info-localssninfo topic=display-info-ogtinfo topic=display-info-opcinfo topic=display-info-
remotessninfo topic=display-info-tcapconninfo
topic=dpc topic=exit topic=export topic=help topic=inbound-gtt
topic=local-endpoint topic=local-endpoint-ip topic=node topic=outbound-gt topic=outbound-gtt
topic=parameters topic=replace-gt topic=route topic=snmp-node topic=target-address
topic=usm-user
SGC[127.0.0.1:10111]> help: topic=conn-ip;
Configuration of IPs for "connection", to make use of SCTP multi-homed feature define multiple IPs for
single "connection".
Object conn-ip is defined by the following parameters:
oname: object name
ip: IP address
conn-name: Name of the referenced connection
Supported operations on conn-ip are listed below ({param=x} - mandatory parameter, [param=x] - optional
parameter):
display-conn-ip: [oname=x],[column=x];
create-conn-ip: {oname=x}, {ip=x}, {conn-name=x};
remove-conn-ip: {oname=x};
SGC[127.0.0.1:10111]>
```

20.3.5 Interactive mode

By default, the CLI starts in interactive mode, which lets the System Administrator execute commands and observe their results through the system terminal. For example, here's a successfully executed command:

```
SGC[127.0.0.1:10111]> display-info-healthinfo: ;
Found 1 object(s):
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|contextExecutio|contextExecutio|allocatedR|allocatedI|workerExecution|workerExecution|nodeId | | |
|executorQu|workerGrou|
|nTime |nCount |eqTasks |ndTasks |Time |Count | |eueSize |pSize |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|0 |0 |0 |0 |0 |0 |CL1-N1 |0 |32 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
SGC[127.0.0.1:10111]>
```

Here's an example of a command that failed:

```
SGC[127.0.0.1:10111]> remove-conn-ip: oname=invalidName;
ERROR REMOVE_OBJECT_FAILED: Instance 'SGC:category=conn-ip,type=m3ua,id=invalidName' doesn't exist.
SGC[127.0.0.1:10111]>
```

20.3.6 Command result truncation

CLI supports the truncation of command result (cell data) to make the output more convenient (In case of large cell data). This configuration property is defined in the `conf/cli.properties` file. Following is the property:

```
table.format.maxCellContentLength=40
```



See also [Supported CLI Operations](#) on page 181

21 Supported CLI Operations

21.1 SGC CLI commands

SGC CLI commands can be grouped into four sets of commands:

- [Management of SGC Stack processing objects](#) on page 181
- [Alarms and Event history](#) on page 184
- [Statistics](#) on page 187
- [Export / Import](#) on page 188





Most CLI operations are executed on the SGC Cluster using JMX management beans. Therefore the CLI requires successful establishment of a JMX connection to the SGC Cluster.

21.1.1 Management of SGC Stack processing objects

Most processing objects that can be managed within the SGC cluster support a set of [CRUD](#) on page commands. Some processing objects can also be enabled or disabled. Command names are a concatenation of the operation name and processing object name; for example: `create-node`, `display-as`, or `remove-conn-ip`. Generic commands are described below. For a detailed description of operations available for a particular processing object type, please see the [SGC CLI built-in help system](#) on page 177 . The commands for managing processing objects are:

Operation	What it's for
Sample use case	
Examples	
<code>create</code>	Creating a new instance of a processing object within the SGC cluster.

	 The create operation requires providing the mandatory parameters for a given SGC processing object; otherwise a validation error will display on the terminal.
	Creating a new connection object, which together with conn-ip defines remote the IP address of an SCTP association.
	<pre>SGC[127.0.0.1:10111]> create-connection: ; ERROR VALIDATION_FAILED: Mandatory parameter 'oname' was not set. ERROR VALIDATION_FAILED: Mandatory parameter 'port' was not set. ERROR VALIDATION_FAILED: Mandatory parameter 'local-endpoint-name' was not set. ERROR VALIDATION_FAILED: Mandatory parameter 'conn-type' was not set.</pre>
remove	Removes an instance of a processing object within the SGC Cluster.
	Removing the conn-ip associated with a connection.
	<pre>SGC[127.0.0.1:10111]> remove-conn-ip: oname=ip1ForConnectionA; OK conn-ip removed.</pre>
modify	Updates the state of a processing object within the SGC Cluster.
	 Can cause validation/operation errors for processing objects, which must be disabled and deactivated before update.
	Updating the application server configuration (modify-as) by increasing the number of maximum pending messages and maximum pending time required.
	Execution of Modify operation on and active application server (as) processing object resulting in a validation error:
	<pre>SGC[127.0.0.1:10111]> modify-as: oname=AS-RC-1, pending-size=1000; ERROR MODIFY_OBJECT_FAILED: com.cts.ss7.common.SGCEXception: Parameter pending-size cannot be modified when bean AS-RC-1 is enabled or active</pre>
display	Display the configuration of SGC Cluster processing objects.



The column parameter can be specified multiple times to restrict displayed columns.


Displaying the attribute values of connection-processing objects.

Successfully executed display operation for all defined connections:

```
SGC[127.0.0.1:10111]> display-connection:; Found 2 object(s): +-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+ | oname |
dependenci|enabled |active |port |local-endpoint-|conn-type |t-ack |t-daud |t-reconnec|
state-maintenan|asp-id |is-ipsp |out-queue-| | |es | | | |name | | | |t |ce-role | | |
size | +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+ |C-CL2-N1 |0 |true |false |30115 |N1-E |SERVER |2 |60 |6 |ACTIVE |null |true |1000 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+ |C-CL2-N2 |1 |true |false |30105 |N1-E |CLIENT |10 |60 |5 |ACTIVE |1 |true |1000 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Successfully executed display operation for a single connection (oname parameter specified):

```
SGC[127.0.0.1:10111]> display-connection: oname=C-CL2-N2; Found 1 object(s):
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+ | oname |dependenci|enabled |active |port |local-endpoint-|conn-type |t-ack |t-daud
|t-reconnec|state-maintenan|asp-id |is-ipsp |out-queue-| | |es | | | |name | | | |t |
ce-role | | |size | +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+ |C-CL2-N2 |1 |true |false |30105 |N1-E |CLIENT |10 |60 |5 |
ACTIVE |1 |true |1000 | +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+
```

	<p>Successfully executed <code>display</code> operation for a single connection with a restricted number of columns (<code>oname</code> and <code>column</code> parameters specified):</p> <pre>SGC[127.0.0.1:10111]> display-connection: oname=C-CL2-N2,column=oname, column=enabled, column=conn-type; Found 1 object(s): +-----+-----+-----+ oname enabled conn-type +-----+-----+-----+ C-CL2-N2 true CLIENT +-----+-----+-----+</pre>
<p>enable</p> <p>disable</p>	<p>Change the "enabled" state of: <code>as</code> , <code>connection</code> , <code>local-endpoint</code> , <code>node</code> , and <code>snmp-node</code> processing object types.</p> <div>  <p>These processing objects can be considered as fully functional only when enabled. Most configuration parameters of such objects cannot be changed when the processing object is enabled.</p> </div>
	<p>Enabling and disabling a connection:</p> <pre>SGC[127.0.0.1:10111]> enable-<TAB_PRESSED> enable-as enable-connection enable-local- endpoint enable-node enable-snmp-node SGC[127.0.0.1:10111]> enable-connection: oname= oname=A-CONN oname=B-CONN SGC[127.0.0.1:10111]> enable-connection: oname=A-CONN; OK connection enabled. SGC[127.0.0.1:10111]> disable-connection: oname=A-CONN; OK connection disabled.</pre>

21.1.2 Alarms and event history

The SGC CLI provides commands for displaying and clearing alarms generated by the SGC Cluster. Alarms raised and subsequently cleared, or notifications emitted, can be reviewed using a separate operation: `display-event-history`. Display operations for alarms let filter expressions be specified for `id`, `name`, and `severity`. Filter expressions can include % wildcard characters at the start, middle, or end of the expression. Also, the `column` parameter can be specified multiple times to restrict presented columns. Below are some examples:

Displaying all active alarms (no filtering criteria specified):

[illegible]


```

|description |id |name |parameters |severity |timestamp |
+-----+-----+-----+-----+-----+-----+
|The node in the|36 |nodefailure |nodeId='Node101|MAJOR |2014-02-10 10:47:35 |
| cluster disapp| | |',failureDescri| | |
|eared | | |ption='Mis... | | |
+-----+-----+-----+-----+-----+-----+
|The node in the|37 |nodefailure |nodeId='Node102|MAJOR |2014-02-10 10:47:37 |
| cluster disapp| | |',failureDescri| | |
|eared | | |ption='Mis... | | |
+-----+-----+-----+-----+-----+-----+

```

Displaying all active alarms with filters:

```

SGC[127.0.0.1:10111]> display-active-alarm: id=36, severity=M%, name=%failure;
Found 2 object(s):
+-----+-----+-----+-----+-----+-----+
|description |id |name |parameters |severity |timestamp |
+-----+-----+-----+-----+-----+-----+
|The node in the|36 |nodefailure |nodeId='Node101|MAJOR |2014-02-10 10:47:35 |
| cluster disapp| | |',failureDescri| | |
|eared | | |ption='Mis... | | |
+-----+-----+-----+-----+-----+-----+

```

Displaying all active alarms with filters and column parameters:

```

SGC[127.0.0.1:10111]> display-active-alarm: severity=M%, name=%failure, column=id, column=description,
column=timestamp;
Found 1 object(s):
+-----+-----+-----+
|id |description |timestamp |
+-----+-----+-----+
|36 |The node in the|2014-02-10 10:47:35 |
| | cluster disapp| |
| |eared | |
+-----+-----+-----+

```

21.1.3 Clearing an active alarm:



Clearing alarms

Any alarm can be cleared by the System Administrator. There are two `clear` operations exposed by the CLI:

- `clear-active-alarm` — clears an active alarm as defined by the mandatory `id` parameter
- `clear-all-alarms` — clears all active alarms.

```
SGC[127.0.0.1:10111]> clear-active-alarm: id=36;
OK alarm cleared.
```

21.1.4 Displaying all registered alarms:



Displaying event history

The `display-event-history` operation is used to display alarm and notification history. The output of this command can be filtered by specifying any of `last`, `time-from`, and `time-to` parameters. Also, the `column` parameter can be specified multiple times to restrict presented columns.

```
SGC[127.0.0.1:10111]> display-event-history: ;
Found 1 object(s):
+-----+-----+-----+-----+-----+-----+
|description|id|name|parameters|severity|timestamp|
+-----+-----+-----+-----+-----+-----+
|The node in the|36|nodefailure|nodeId='Node102'|MAJOR|2014-02-10 10:47:35|
| cluster disapp|'|',failureDescri|'|
|eared|'|option='Mis...|'|
+-----+-----+-----+-----+-----+-----+
```

21.2 Statistics (Info)

A set of commands allow interrogation of statistics exposed by the SGC Stack. For details, please see [Statistics](#) on page 149 . The available statistics are:

Module	Statistic
M3UA	asinfo
	associationinfo
	dpcinfo
SCCP	localssninfo
	ogtinfo
	pcinfo
	remotessninfo
TCAP	tcapconninfo
TOP	healthinfo



Filtering statistical information

Commands displaying statistical information support filtering. Filtering of statistics is based on the equality between the filter value and statistic column value. Also, the `column` parameter can be specified multiple times to restrict presented columns.

Below are some examples.

21.2.1 Displaying statistic without filters:

```
SGC[127.0.0.1:10111]> display-info-asinfo;;
Found 1 object(s):
+-----+-----+-----+-----+-----+-----+
|connectionId |asId |TXCount |RXCount |status |nodeId |
+-----+-----+-----+-----+-----+-----+
|C-CL2-N1 |AS-RC-1 |0 |0 |INACTIVE |CL1-N1 |
+-----+-----+-----+-----+-----+-----+
```





21.2.2 Displaying asinfo statistics with filters on the nodeId:

```
SGC[127.0.0.1:10111]> display-info-asinfo:<TAB_PRESSED>
RXCount TXCount asId column connectionId nodeId status
SGC[127.0.0.1:10111]> display-info-asinfo: nodeId=CL1-N1;
Found 1 object(s):
+-----+-----+-----+-----+-----+-----+
|connectionId |asId |TXCount |RXCount |status |nodeId |
+-----+-----+-----+-----+-----+-----+
|C-CL2-N1 |AS-RC-1 |0 |0 |INACTIVE |CL1-N1 |
+-----+-----+-----+-----+-----+-----+
```

21.3 Export / Import

The following commands allowing exporting and importing the SGC Stack configuration to or from a text file.

Operation	What it's for
Examples	
export	Produces a dump of the current configuration in a format that is directly usable by the batch command.

	<div data-bbox="474 204 527 250"></div> <div data-bbox="562 215 1612 245">When the file argument is not provided, the output will be printed to the standard location.</div> <div data-bbox="474 321 527 367"></div> <div data-bbox="562 316 1850 378">The export operation can be used also in non-interactive mode, if the <code>sgc-cli.sh</code> script is executed with the <code>-x, --export</code> parameter.</div> <div data-bbox="457 451 722 480">Non-interactive mode:</div> <div data-bbox="474 521 1001 550"><pre>./sgc-cli.sh -x config-backup.mml</pre></div> <div data-bbox="457 589 667 618">Interactive mode:</div> <div data-bbox="474 659 1730 688"><pre>SGC[127.0.0.1:10111]> export: file=config-backup.mml OK configuration exported.</pre></div>
batch	<div data-bbox="457 737 1881 808">Loads a text file containing a set of MML commands, and executes them after connecting to the SGC Cluster (in a format that is produced by the <code>export</code> command).</div> <div data-bbox="474 862 527 907"></div> <div data-bbox="562 857 1843 919">Batch operation can also be used in non-interactive mode, if the <code>sgc-cli.sh</code> script is executed with the <code>-b, --batch</code> parameter.</div> <div data-bbox="474 979 527 1024"></div> <div data-bbox="562 972 1801 1063">The batch command supports an additional flag: <code>-stopOnError, --stopOnError</code>. If this flag is set to false, it won't stop processing commands even if execution of a command results in an error. (The default behaviour is to stop processing commands after the first error.)</div> <div data-bbox="457 1135 722 1164">Non-interactive mode:</div> <div data-bbox="474 1205 1304 1234"><pre>./sgc-cli.sh -b config-backup.mml -stopOnError false</pre></div> <div data-bbox="457 1273 667 1302">Interactive mode:</div> <div data-bbox="474 1343 1860 1398"><pre>SGC[127.0.0.1:10111]> batch: file=set-of-displays.mml display-conn-ip: oname=conn-ip1; Found 1 object(s): +-----+-----+-----+-----+ oname</pre></div>

```
|dependenci|ip |conn-name | | |es | | | +-----+-----+-----+
+-----+ |conn-ip1 |0 |192.168.1.101 |C-CL2-N1 | +-----+-----+
+-----+-----+ display-conn-ip: oname=conn-ip2; Found 1 object(s):
+-----+-----+-----+-----+ |oname |dependenci|ip |
conn-name | | |es | | | +-----+-----+-----+-----+
|conn-ip2 |0 |192.168.1.192 |C-CL2-N1 | +-----+-----+-----+
+-----+
```

22 SGC TCAP Stack

22.1 What is the SGC TCAP Stack ?

The SGC TCAP Stack is a Java component that is embedded in the CGIN Unified RA and the IN Scenario Pack to provide OCSS7 support. The TCAP stack communicates with the SGC via a proprietary protocol.



For a general description of the TCAP Stack Interface defined by the CGIN Unified RA, please see [Inside the CGIN Connectivity Pack](#).

22.2 TCAP Stack and SGC Stack cooperation

After the CGIN Unified RA is activated the SGC TCAP stack uses one of two procedures to establish communication with the SGC stack:

1. The newer, and recommended, `ocss7.sgcs` registration process, introduced in release 1.1.0.
2. The legacy `ocss7.urlList` registration process, supported by releases up to and including 1.1.0.

22.2.1 'ocss7.sgcs' Connection Method

This connection method:

- [automatically load balances](#) on page 192 all traffic between all available SGCs; and
- [introduces dialog failover](#) on page 192 when an SGC becomes unavailable for any reason.

When properly configured, all TCAP stacks in the Rhino cluster will be connected to all available SGCs in the OCSS7 cluster simultaneously, with every connection servicing traffic. This is sometimes referred to as the Meshed Connection Manager because of the fully meshed network topology it uses.

TCAP Stack registration process

After the CGIN Unified RA is activated, the SGC TCAP stack connects to all configured SGC(s). For each connection the TCAP stack will:

1. Perform a basic handshake with the SGC over the connection, negotiating the data protocol version to be used. If the SGC does not support the requested protocol version it will respond with handshake failure, closing the connection. If the SGC does support the requested protocol version it will respond with a handshake success message.
2. Send an extended handshake request to the SGC. This handshake includes a unique TCAP stack identifier as well as any extended features supported by the TCAP stack. At the time of writing only one feature exists: `PREFIX_MIGRATION` ; see [Prefix Migration](#) on page 192 for details.
3. The SGC assigns a TCAP local transaction ID prefix for this connection.
4. The SGC creates local state to manage the data transfer with the TCAP stack, and updates the cluster state with information about the newly connected TCAP stack.
5. The SGC sends an extended handshake response to the TCAP stack, including the local transaction prefix assigned as well as indicating any extended features supported by the TCAP stack. At the time of writing, the only such feature is `PREFIX_MIGRATION` .

At this point, the TCAP stack is operational and can both send and receive TCAP traffic.

Load balancing TCAP stack data

Dialogs are load balanced across all active connections between the SGCs and the TCAP stacks. An SGC that receives traffic from the SS7 network will round-robin new dialogs between all TCAP stacks that have an active connection to it. Similarly, each TCAP stack will round-robin new outgoing dialogs between the SGCs it is connected to.

If a configured connection is down for any reason the TCAP stack will attempt to re-establish it at regular intervals until successful.

In some OA&M or failure situations an SGC may have no TCAP stack connections. When this happens the SGC will route traffic to the other SGCs in the cluster for forwarding to the TCAP stacks. Traffic will remain balanced across the TCAP stacks.

If a new SGC needs to be added to the cluster with a new connection address then the TCAP stack configuration will need to be adjusted. This adjustment can be done either before or after the SGC is installed and configured, and the cluster will behave as described above. Updates to the TCAP stack `ocss7.sgcs` list can be done while the system is active and processing traffic — existing connections will not be affected unless they are removed from the list, and the TCAP stack will attempt to establish any newly configured connections.

Prefix Migration

This feature provides dialog failover for in-progress dialogs when an SGC becomes unavailable to a TCAP stack for some reason.



This feature does not provide failover between Rhino nodes when a Rhino node becomes unavailable for some reason. Failover between Rhino nodes is not available in CGIN 1.5.3 with any TCAP stack because CGIN itself does not provide replication facilities.

As detailed in [TCAP Stack registration process](#) on page 191, each connection to an SGC is allocated a TCAP local transaction ID prefix. Under normal operating conditions, each TCAP dialog continues to use the same connection between a TCAP stack and an SGC for its lifetime, with the connection identified by the prefix the connection was assigned. If, however, an SGC becomes unavailable the Prefix Migration feature allows the prefixes for all affected connections to be distributed to the other SGCs in the cluster and attached to suitable existing TCAP stack connections (if available). Once prefix migration has completed, existing dialogs will continue via an existing connection to a different SGC.

For example, TCAP stack A is connected to SGCs 1 and 2. It has a different prefix for each connection; A1 and A2 respectively. If SGC 2 terminates unexpectedly, the prefix associated with A2 will be migrated to connection A1. All traffic received by the SGCs for that prefix will now be sent to the TCAP stack via connection A1, which is hosted on SGC 1.

It is important to note that prefixes migrate between SGCs, not TCAP stacks. For prefix migration to work there must be an existing connection between the original TCAP stack and at least one substitute SGC which is capable of accepting a new migrated prefix. An SGC may not be capable of accepting a prefix migration because of configuration and resource limitations (see below). Prefixes which cannot be allocated to a replacement SGC will be dropped and their in-flight dialogs will be lost.

Each active prefix in an SGC has associated resource overhead so migrated prefixes are only used for existing dialogs. New dialogs will not be created on a migrated prefix. Once the last dialog ends on a migrated prefix that prefix and associated resources are released.

Please note that prefix migration support requires that the SGC's heap is configured appropriately. Please see the `sgc.tcap.maxMigratedPrefixes` configuration property documentation in [Static SGC instance configuration](#) on page 81 for information on the amount of heap required to support prefix migration.

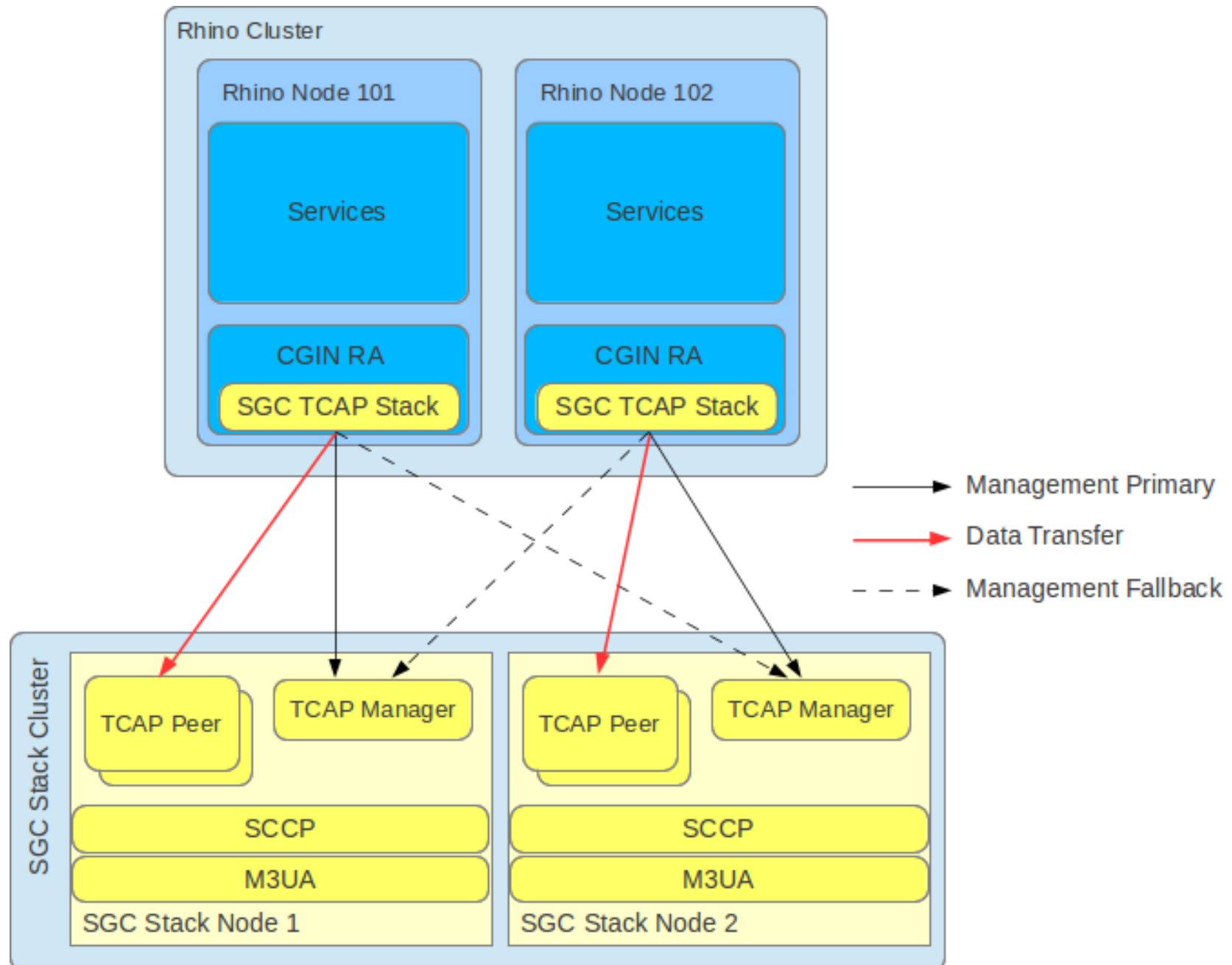


Failure recovery can never be perfect — network messages currently being processed at the time of failure will be lost — so some dialogue failure must be expected.

22.2.2 Legacy 'ocss7.urlList' Connection Method

This is the method used by the TCAP stack and SGCs prior to release 1.1.0. This method is still available to ensure backwards compatibility between TCAP stacks and the SGC. It is strongly recommended that all new installations use the new `ocss7.sgcs` method instead as this provides failover capabilities as well as automatic load balancing without the need for intervention.

Below is a high-level overview diagram, followed by descriptions, concentrating on components directly involved in TCAP Stack and SGC Stack cooperation. SGC Stack components are the items colored yellow ; components colored blue (and similar) are provided by Rhino platform or are part of Operating System environment.



- **TCAP Manager** is the software component that the TCAP Stack queries initially, to register within the SGC Stack and establish a data transfer connection.
- **TCAP Peer** is the software component within the SGC Stack Node that manages the data transfer connection with the TCAP Stack.



The TCAP Manager listen address is configured as part of configuring the SGC Node. For details, please see the `stack-http-address` and `stack-http-port` attributes in [General Configuration](#) on page 93 .

TCAP Stack registration process

After the CGIN Unified RA is activated, the SGC TCAP Stack goes through a two-phase process to establish communication with the SGC Stack:

1. Assigning to the connection to the TCAP Manager an SGC Node, where the data transfer connection should be established.
2. Connecting to SGC Stack Node, to register it within the SGC Cluster, and establish the data transfer connection.

Detailed Stack registration steps

1. The TCAP Stack connects to the first active TCAP Manager from a user-provided list of TCAP Manager addresses (corresponding to the nodes of the SGC Stack Cluster).
 2. After the connection with the TCAP Manager is established, the TCAP Stack sends a request containing information about the [SSN](#) on page represented by the TCAP Stack.
 3. The TCAP Manager selects an SGC Node within the cluster that will be responsible for managing the data transfer connection with the TCAP Stack.
 4. The address where the selected SGC Node listens for data transfer connections is sent back as part of the response to the TCAP Stack.
-
1. The TCAP Stack connects to the selected SGC Node and begins registration by sending a handshake message containing its SSN and version of the data transfer protocol that it supports (this is a custom protocol used to exchange TCAP traffic between the TCAP Stack and the SGC Node; both v1.0.0 and v2.0.0 versions are supported in this release of OCSS7).
 2. The SGC Node assigns a TCAP local transaction prefix to the TCAP Stack,
 3. The SGC Node creates a TCAP Peer responsible for managing the data transfer connection with the TCAP Stack, and updates the cluster state with information about the newly connected TCAP Stack.
 4. The TCAP local transaction prefix is sent to the TCAP Stack as part of a successful handshake response.

At this stage, the TCAP Stack is operational and can both originate and receive TCAP traffic.



How the TCAP Manager selects an SGC Node

The goal of the node-selection process is to load balance the number of connected TCAP Stacks between SGC Nodes within an SGC Cluster. The TCAP Manager does that by following a very simple algorithm:

1. If the IP address used by the TCAP Stack is the same as the IP address used by one of the nodes for TCAP data transfer connections, select that SGC node (to minimise network latency).
2. Otherwise, select the node with the least number of connected TCAP Stacks.
3. Otherwise (if all nodes have the same number of connected TCAP Stacks) select any SGC Node.



When Phase 2 registration fails...

The TCAP Stack can fail to register with the SGC Node. Possible reasons include:

- **handshake timeout** — the process times out waiting for a handshake between the SGC and TCAP stack
- **resource limitation** — the installation reaches the maximum number of TCAP peers/clients connected to this SGC instance (see [Configuring the SS7 SGC Stack](#) on page 81)
- **out of prefixes** — there is no free local transaction prefix that could be assigned to the TCAP Stack
- **invalid version** — the process is using an unsupported version of the data transfer protocol.

To see why registration fails, check the log file.

Data transfer connection failover

After the TCAP Stack detects data transfer connection failure, it repeats the registration process by attempting to establish a connection to one of the configured TCAP Managers. Connection attempts are in order of TCAP Manager addresses defined by the `ocss7.urlList` property. This process does not differ from the standard TCAP Stack registration process, as TCAP Manager selects one of the available SGC Nodes. For proper failover behaviour, the TCAP Stack should be configured with the TCAP Manager addresses of all nodes within the SGC Cluster.

Rebalancing TCAP Stack data transfer connections

When an SGC Node joins a running SGC Cluster (for example after a planned outage or failure), the already established TCAP Stack data transfer connections are NOT affected. That is, existing TCAP Stack data transfer connections are NOT rebalanced between SGC Nodes.



This example uses a basic production deployment of an SGC Cluster, composed of two SGC Nodes cooperating with the CGIN Unified RA, deployed within a two-node Rhino cluster (as depicted above).

Imagine the following scenario:

- Due to hardware failure, SGC Stack Node 2 was not operational. This resulted in both instances of the CGIN Unified RA being connected to SGC Stack Node 1.
- That hardware failure is removed, and SGC Stack Node 2 is again fully operational and part of the cluster.
- To rebalance the data transfer connections of the CGIN Unified RA entity running within Rhino cluster, that entity must be deactivated and activated again on one of the Rhino nodes. (Deactivation is a graceful procedure where the CGIN Unified RA waits until all dialogs that it services are finished; at the same time all new dialogs are directed to the CGIN Unified RA running within the other node.)

You would do the following:

1. Make sure that the current traffic level is low enough so that it can be handled by a CGIN Unified RA on a single Rhino node.
2. Deactivate the CGIN Unified RA entity on one of the Rhino nodes.
3. Wait for the deactivated CGIN Unified RA entity to become `STOPPED`.
4. Activate the previously deactivated CGIN Unified RA entity,



Per-node activation state

For details of managing per-node activation state in a Rhino Cluster, please see [Per-Node Activation State](#) in the *Rhino Administration and Deployment Guide*.

22.3 TCAP Stack configuration

General description and configuration of the CGIN RA is detailed in the CGIN RA Installation and Administration Guide. Below are configuration properties specific to the SGC TCAP Stack.

Parameter	Usage and description	Active Reconfig?
Values		
Default		
stack	Short name of the TCAP stack to use.	✗
	for the SGC TCAP Stack, this value must be <code>ocss7</code>	
ocss7.schThreads	Maximum number of threads used by the scheduler. Number of threads used to trigger timeout events.	✗
	in the range 1 - 100	
	10	
ocss7.schNodeListSize	Number of events that may be scheduled on a single scheduler thread.	✗
	in the range 1000 - 2000000 or 0 (autosize) ocss7.schNodeListSize * ocss7.schThreads should be directly proportional to ocss7.trdpCapacity (the maximum number of open dialogs)	
	0	

ocss7.taskdpCapacity	Maximum number of inbound messages and timeout events that may be waiting to be processed.	✗
	in the range 1000 - 2000000 or 0 (autosize) ocss7.taskdpCapacity should be directly proportional to ocss7.trdpCapacity (the maximum number of open dialogs)	
	0	
ocss7.trdpCapacity	Maximum number of opened transactions (dialogs).	✗
	in the range 1000 - 2000000	
	100000	
ocss7.wgQueues	Number of threads used by the worker group to process timeout events and inbound messages.	✗
	in the range 1 - 100	
	100	
ocss7.wgQueuesSize	Maximum number of tasks in one worker queue.	✗
	in the range 1000 - 2000000 or 0 (autosize) ocss7.wgQueues * ocss7.wgQueuesSize should be directly proportional to 2 * ocss7.taskdpCapacity (the maximum number of triggered timeout and inbound messages)	
	0	

ocss7.senderQueueSize	Maximum number of outbound messages in the sender queue.	✗
	in the range 1000 - 2000000 or 0 (autosize)	
	(if less than default value the default value is silently used)	
	ocss7.senderQueueSize is directly proportional to ocss7.trdpCapacity (the maximum number of open dialogs)	
ocss7.sgcs	0	✓ *
	Comma-separated list of SGCs to connect to. Only one of ocss7.sgcs or ocss7.urlList may be configured at a time, and this is the recommended one - see TCAP Stack and SGC Stack cooperation on page 191 for details.	
	comma-separated list in the format address:port , pointing to the listening addresses of SGC data ports within the SGC stack cluster	
ocss7.urlList	unset	✓ *
	Comma-separated list of URLs used to connect to legacy TCAP Manager(s). Only one of ocss7.sgcs or ocss7.urlList may be configured at a time, and ocss7.sgcs is the recommended one - see TCAP Stack and SGC Stack cooperation on page 191 for details.	
	comma-separated list in the format address:port , pointing to the listening addresses of TCAP Managers within the SGC Stack cluster	
ocss7.urlList.retrySleep	unset	✓
	Wait interval in milliseconds between subsequent connection attempts for the ocss7.urlList. in the range 0 - 600000	

	not related to single URLs on the list (but the whole list)	
	1000	
ocss7.local-ssn	SSN number used when the local-sccp-address property does not provide a SSN (for example, if set to auto).	✗
	in the range 2 - 255	
ocss7.heartbeatEnabled	v1.0.0 protocol variant only - Enables or disables the OCSS7 to SGC connection heartbeat. N.B. if the heartbeat is enabled on the SGC it <i>must</i> also be enabled here. v2.0.0 protocol variant - heartbeat is always enabled.	✗
	true or false	
	true	
ocss7.heartbeatPeriod	The period between heartbeat sends in seconds. Heartbeat timeout is also a function of this value, as described in Data Connection Heartbeat Mechanism on page 204 . The value configured here <i>must</i> be smaller than the value of <code>com.cts.ss7.commssp.server.recv Timeout</code> given to the SGC on page 81 .	✗
	2 or larger	
	5	

* If a TCAP data transfer connection is established, changing this property has no effect until the data transfer connection fails and the TCAP Stack repeats the registration process.

22.4 SGC-TCAP stack protocols

Version 1.1.0 of OCSS7 supports two proprietary data protocols; the original v1.0.0 protocol and a new v2.0.0 protocol. The data protocol version is automatically negotiated and is not user configurable.

22.4.1 v2.0.0

This is a protobuf-based protocol, running over TCP/IP. It was introduced in OCSS7 1.1.0.

Handshake

v2.0.0 uses a two-stage handshake. The first stage is the original proprietary basic handshake, used to agree on a protocol version to be used for TCAP stack - SGC communications.

Once the basic handshake has successfully negotiated the v2.0.0 protocol the TCAP stack and SGC both switch to using this protocol for further communications.

Next, the TCAP stack will send an extended handshake to the SGC. This handshake includes:

1. A list of extended features supported by the TCAP stack. At present `PREFIX_MIGRATION` is the only extended feature available.
2. The heartbeat period configured in the TCAP stack (TCAP stack property `ocss7.heartbeatPeriod`)
3. An identifier that uniquely identifies this TCAP stack (this identifier will be the same for all connections originating from this TCAP stack)

The SGC will determine if TCAP stack connection can be accepted, and if it can, will send an extended handshake success response which includes:

1. A list of extended features supported by the SGC. At present `PREFIX_MIGRATION` is the only extended feature available.
2. The prefix assigned to this connection, to be used to allocate transaction IDs for any new dialogs sent/received over this connection.

If the TCAP stack connection cannot be accepted, the SGC will send an extended handshake response which includes:

1. The reason why the TCAP stack connection could not be accepted

Heartbeat

The data connection between the TCAP stack and SGC features a configurable heartbeat consisting of:

1. A heartbeat request/response message pair
2. TCAP stack side timeout detection
3. SGC side timeout detection

The heartbeat period is configured in the TCAP stack by setting the TCAP stack `ocss7.heartbeatPeriod` to the desired value (in ms), and automatically communicated to the SGC by the TCAP stack during the extended handshake.

The TCAP stack will send a `HeartbeatPing` message at the configured interval. On receipt of a `HeartbeatPing` the SGC will send a `HeartbeatPong`.

If the SGC does not receive a `HeartbeatPing` message within $2 * \text{ocss7.heartbeatPeriod}$ ms of the last `HeartbeatPing` or the extended handshake completing, it will mark the heartbeat as having failed and close the connection to the TCAP stack.

If the TCAP stack does not receive a `HeartbeatPong` message before sending its next ping (i.e. within `ocss7.heartbeatPeriod` ms) it will mark the heartbeat as having failed and close the connection to the TCAP stack.

22.4.2 v1.0.0

This is the original proprietary protocol used by OCSS7 version 1.0.0.x and 1.0.1.x. OCSS7 1.1.0.x also supports this protocol version to ensure backwards compatibility with older clients/servers.

Handshake

This protocol uses a basic handshake to establish the connection between TCAP stack and SGC. This handshake includes a protocol version field, which will be set to `v1.0.0`. If the SGC cannot accept this connection, for example, unsupported protocol version, insufficient resources available, it will send a handshake failure response.

If the SGC can accept this connection it will send a handshake success response which includes the prefix allocated for this connection.

Data Connection Heartbeat Mechanism

The data connection between the TCAP stack and SGC supports a heartbeat mechanism that can be used to detect failed connections in a timely manner. This mechanism consists of three components:

1. A heartbeat request/response message pair
2. TCAP stack side timeout detection
3. SGC side timeout detection

The heartbeat request/response message pair is enabled by setting the TCAP stack `ocss7.heartbeatEnabled` property to `true`. This configures the TCAP stack to send a heartbeat request. The SGC will respond to any heartbeat request with a heartbeat response message, regardless of SGC configuration. The frequency with which the heartbeat request message is sent (and therefore also the heartbeat response) is configured with the TCAP stack's `ocss7.heartbeatPeriod` configuration parameter.

If heartbeats are enabled in the TCAP stack, then the TCAP stack will automatically perform timeout detection. If a heartbeat response message hasn't been seen within a period of time equal to twice the heartbeat period ($2 * \text{ocss7.heartbeatPeriod}$) the connection will be marked as defective and closed, allowing the TCAP stack to select a new SGC data connection.

The SGC stack may also perform timeout detection; this is controlled by the [SGC properties](#) on page 81 `com.cts.ss7.commsp.heartbeatEnabled` and `com.cts.ss7.commsp.server.recvTimeout`. When the `heartbeatEnabled` property is set to `true` the SGC will close a connection if a heartbeat request hasn't been received from that TCAP stack in the last `recvTimeout` seconds.

It is important to note that if the SGC stack is configured to perform timeout detection then every TCAP stack connecting to it must also be configured to generate heartbeats. If a heartbeat-disabled TCAP stack connects to a heartbeat-enabled SGC the SGC will close the connection after `recvTimeout` seconds, resulting in an unstable TCAP stack to SGC connection.

23 Upgrading the SGC and TCAP stack

This section of the manual covers upgrading the SGC and the TCAP stack.

23.1 Upgrading from OCSS7 1.0.1.x to OCSS7 1.1.0.x

23.1.1 Notable Changes

The TCAP stack now supports a meshed-style connection architecture, where each TCAP stack instance is connected to one or more SGCs simultaneously. The meshed connection style also enables prefix migration between SGCs. That is, if an SGC responsible for a prefix terminates for any reason, another SGC can take on responsibility for it, directing messages for that prefix towards the correct TCAP stack.

Meshed-style connections require the use of the new `ocss7.sgcs` configuration property. The old `ocss7.urlList` property continues to be available for backwards compatibility reasons, but it is highly recommended to switch to the meshed-style architecture for maximum resiliency.

See [ocss7.sgcs Connection Method](#) on page for more details.

The SGCs have a new configuration property `sgc.tcap.maxMigratedPrefixes`. Similar to the existing `sgc.tcap.maxPeers` property, this controls the maximum number of prefixes that may be migrated to any given SGC and must be configured appropriately. See [Static SGC instance configuration](#) on page 81 for more details.

An SGC cluster may contain only 1.0.1.x members or only 1.1.0.x members. A cluster consisting of 1.0.1.x and 1.1.0.x SGC nodes will have undefined behaviour and is not supported.

23.1.2 Online Upgrade Using STP Redirection

This approach manages the upgrade externally to Rhino and OCSS7, and requires support from the STP and surrounding network, as well as support in the existing service. It can be used for all types of upgrade.

Prerequisites

Before upgrading using STP redirection, make sure that:

- Inbound TC-BEGINs are addressed to a "logical" GT. The STP translates this GT to one-of-N "physical" addresses using a load-balancing mechanism. These "physical" addresses route to a particular cluster.
- The STP needs the ability to reconfigure the translation addresses in use at runtime.
- The old and new clusters are assigned different "physical" addresses.
- The service ensures that initial responding TC-CONTINUEs provide an SCCP Calling Party Address that is the "physical" address for the cluster that is responding.
- Subsequent traffic uses the "physical" address using normal TCAP procedures.

Upgrade process

To upgrade using STP redirection:

1. Set up the new clusters (Rhino and SGC) with a new "physical" address. Ensure that the new Rhino cluster has a different clusterID to the existing Rhino cluster. Similarly, the new SGC cluster must have a different Hazelcast cluster ID to the existing SGC cluster.
2. Configure and activate the new clusters.
3. Reconfigure the STP to include the new cluster's physical address when translating the logical GT.
4. Verify that traffic is processed by the new cluster correctly.
5. Reconfigure the STP to exclude the old cluster's physical address when translating the logical GT.
6. Wait for all existing dialogs to drain from the old clusters.
7. Halt the old clusters.

24 Glossary

AS

Application Server

CLI

Command Line Interface

CPC

Concerned Point Code

CRUD

Create Remove Update Display

DPC

Destination Point Code

GT

Global Title

IPSP

IP Server Process

JDK

Java Development Kit

JKS

Java KeyStore

MML

Man Machine Language

SNMP

Simple Network Management Protocol

SG

Signalling Gateway

SPC

Signalling Point Code

SS7 SGC

Signalling System No. 7 Signalling Gateway Client

SSL

Secure Socket Layer

SSN

SubSystem Number

USM

User Security Model