Oracle® Communications Service Broker

Signaling Domain Configuration Guide

Release 6.0

E23526-02

March 2012

ORACLE®

Oracle Communications Service Broker Signaling Domain Configuration Guide, Release 6.0

E23526-02

Contents

3 Configuring the SS7 Signaling Server Unit for TDM

4 Configuring SIP Signaling Server Units

5 Configuring Diameter Signaling Server Units

6 Configuring RADIUS Signaling Server Units

7 Configuring PCP Signaling Server Units

8 Configuring SMPP Signaling Server Units

9 Configuring the Web Services Signaling Server Unit

Preface

This document provides reference information on configuring Oracle Communications Service Broker Signaling Domain using the Administration Console and Java MBeans.

Audience

This document is intended for system administrators who are responsible for configuring Service Broker in their network.

This document assumes that the reader is already familiar with:

- Signaling System #7 (SS7) – both SIGTRAN and TDM
- Session Initiation Protocol (SIP)
- Diameter protocol
- Java Management Extensions (JMX)

Related Documents

The following documents provide additional information about Service Broker and relevant standards.

- *Oracle Communications Service Broker Release 6.0 Concepts Guide*
- *Oracle Communications Service Broker Release 6.0 Installation Guide*
- *Oracle Communications Service Broker Release 6.0 System Administrator's User's Guide*
- *Oracle Communications Service Broker Release 6.0 Processing Domain Configuration Guide*
- *Oracle Communications Service Broker Release 6.0 Online Mediation Controller Implementation Guide*
- *Oracle Communications Service Broker Release 6.0 Policy Controller Implementation Guide*
- *Oracle Communications Service Broker Release 6.0 Subscriber Store User's Guide*
- *Oracle Communications Service Broker Release 6.0 Release Notes*

Downloading Oracle Communications Documentation

Oracle Communication Service Broker documentation is available from the Oracle software delivery Web site:

http://edelivery.oracle.com

Additional Oracle Communications documentation is available from Oracle Technology Network:

http://www.oracle.com/technetwork/index.html

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

About Signaling Domain Configuration

This chapter provides an overview of Signaling Domain configuration.

About the Signaling Domain

A Signaling Domain is a set of servers, known as Signaling Servers, on which you install Signaling Server Units (SSUs). Service Broker uses SSUs to communicate with a network.

Service Broker provides different types of SSUs. Each type supports a certain protocol that allows Service Broker to communicate with the following networks:

- SSU for SS7 networks in which traffic is carried out over SIGTRAN M3UA ("Configuring the SS7 Signaling Server Unit for SIGTRAN" for more information on SIGTRAN SSUs configuration)
- SSU for SS7 networks in which traffic is carried out over TDM (see "Configuring the SS7 Signaling Server Unit for TDM" for more information on TDM SSUs configuration)
- SSU for SIP networks (see "Configuring SIP Signaling Server Units" for more information on SIP SSU configuration)
- SSU for Diameter networks (see "Configuring Diameter Signaling Server Units" for more information on Diameter SSU configuration)
- SSU for RADIUS networks (see "Configuring RADIUS Signaling Server Units" for more information on RADIUS SSU configuration)
- SSU for communicating with Short Message System Centers (SMSCs) through the SMPP protocol (see "Configuring SMPP Signaling Server Units" for more information)
- SSU for communication with the Oracle BRM application through the Portal Communications Protocol (see "Defining PCP Network Entities" for more information on PCP SSU configuration)
- SSU for communication with external entities using SOAP or REST over HTTP (see "Configuring the Web Services Signaling Server Unit" for more information)

Depending on your specific requirements, you can group Signaling Servers into groups and dedicate each server group to a specific type of the SSU. In this case, each group of Signaling Servers provides access to a different network. Alternatively, you can deploy different SSUs—for example, SIP SSU and Diameter SSU—on Signaling Servers of the same group.

About the Configuration Process

During the configuration process, you define how an SSU handles traffic received from a network and to which interworking modules the SSU forwards this traffic for further processing. In addition, you specify how an SSU sends traffic from interworking modules to a network.

You need to configure each SSU deployed in the domain separately.

You configure SSUs using one of the following methods:

- Administration Console, which provides a graphical user interface
- Java MBeans, which expose a set of attributes and operations that you use to configure SSUs programmatically

Each chapter of this guide is dedicated to a specific type of an SSU and describes both configuration methods.

2

Configuring the SS7 Signaling Server Unit for SIGTRAN

This section describes how to configure SS7 SSU in a network in which SS7 traffic is carried over SIGTRAN M3UA

Configuring SS7 SSU for SIGTRAN with the Administration Console

This section describes how to configure SS7 SSU for SIGTRAN using the Service Broker Administration Console.

The section provides graphical representations of a deployment example.

To access the SS7 SSU configuration pane:

1. In the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Select **SSU SS7 SIGTRAN**.

The SSU SS7 SIGTRAN configuration pane contains the tabs described in Table 2–1.

> **Note:** You must configure the parameters exactly in the order they are presented in Table 2–1.

Table 2–1 M3UA Configuration Tabs

Tab	Description
SSU SS7 SIGTRAN	Enables you to assign a point code to a Service Broker SSU and define the underlying SS7 stack. See "SSU SS7 SIGTRAN" for more information.
M3UA	Enables you to configure the M3UA layers of the SS7 stack. See "M3UA" for more information.
SCCP	Enables you to configure SCCP addresses: subsystems and global titling. See "SCCP" for more information.
Routing	Enables you to define how the SS7 SSU routes incoming SS7 messages to internal Service Broker IMs. See "Routing" for more information.

SSU SS7 SIGTRAN

The SSU SS7 SIGTRAN tab enables you to assign a point code to a Service Broker SSU and configure the M3UA stack runtime options.

To access the SSU SS7 SIGTRAN tab:

- In the SSU SS7 SIGTRAN configuration pane, click the **SSU SS7 SIGTRAN** tab.

  The **General** subtab contains the parameters described in Table 2–2.

Table 2–2 SS7 SSU SIGTRAN Parameters

Name	Type	Description
Vendor	STRING	Specifies the SIGTRAN stack vendor. Possible options: ■ isigtran ■ dialogic
Standard	STRING	Specifies which standard to use to encode M3UA messages. Possible values: ■ ANSI ■ ETSI Default value: ETSI
SS7 Stack IP	INT	The IP address where the SS7 process (that is, the SS7 stack wrapper) is running.
SS7 Stack Port	INT	The port that the SS7 process is using to listen to messages from the SS7 SSU. This is the same port you specify to the SS7 process, in the command line, when you start it. See "Starting and Stopping the SS7 Process" in *Service Broker System Administrator's Guide*.

M3UA

The M3UA tab enables you to configure the M3UA layers of the SS7 stack.

To access the M3UA tab:

1. In the SSU SS7 configuration pane, click the **M3UA** tab.
2. In the **SSU Instance** list, select the Signaling Server that you want to configure.
3. Select one of the subtabs described in Table 2–3.

Table 2–3 M3UA Subtabs

Subtab	Description
Local Point Code	Enables you to specify a point code for each SSU instance. See "Local Point Code" for more information.
Connectivity	Enables you to set up an IP connection between the Service Broker SSU instances and an SS7 network. See "Connectivity" for more information.
Network Mapping	Enables you to define SCTP associations and connect SSUs to adjacent signaling points. See "Network Mapping" for more information.

Table 2–3 (Cont.) M3UA Subtabs

Subtab	Description
Network Routing	Enables you to configure routes to entities in an SS7 network. See "Network Routing" for more information.

Local Point Code

The Local Point Code subtab enables you to specify a point code of the SSU instance that you selected in the SSU Instance list, as described in Table 2–4.

Table 2–4 Point Code Field

Name	Type	Description
Local Point Code	INT	Specifies a local point code of the SSU instance that you selected in the SSU Instance list. A value of the parameter must be integer.

Connectivity

The Connectivity subtab enables you to set up an IP connection between the Service Broker SSU instances and an SS7 network. You configure SSU instances as local systems and other SS7 network entities that are directly connected to the SSU instance as remote systems.

Table 2–5 describes the subtabs on the SS7 SSU Connectivity subtab.

Table 2–5 SS7 Connectivity Subtab

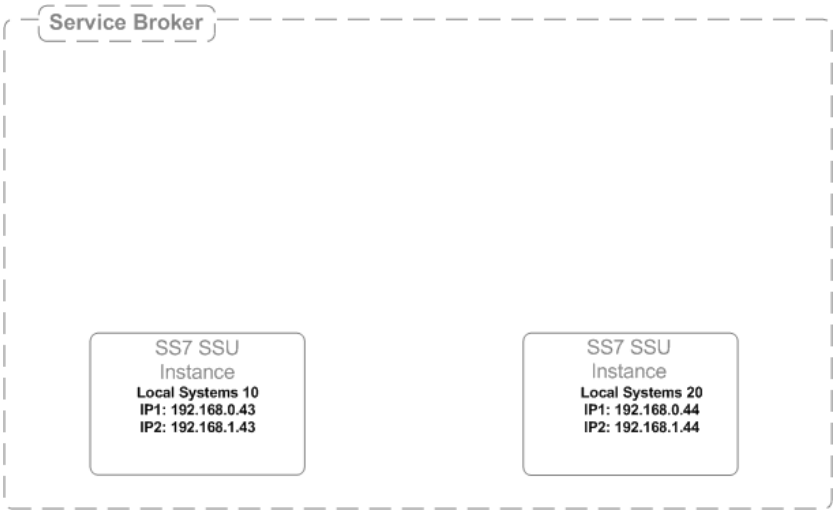
Subtab	Description
Local System	Enables you to configure the SS7 SSU instance as a local M3UA system. See "Configuring the Local System" for more information.
Remote Systems	Enables you to configure network entities. See "Configuring Remote Systems" for more information.

Configuring the Local System

The Local System subtab enables you to configure the SS7 SSU instance as a local M3UA system.

Figure 2–1 shows an example of configuration of the local systems components.

Figure 2–1 Configuration Example: M3UA Local Systems



The Local System subtab contains a table in which you configure one row that defines an SSU instance as a local system. When defining the SSU instance as a local system, you need to specify the fields described in Table 2–6.

Table 2–6 Local Systems Fields

Name	Type	Description
Name	STRING	Specifies a descriptive name for the local system
Routing Context	INT (11)	Specifies a unique identifier that logically identifies a local system when communicating with a traditional SS7 network through a signaling gateway. Routing Context can be set to any value between 0 and 2147483647. Default value: 0.
SS7 Mode	STRING	Specifies an SS7 signaling mode that determines the type of SS7 traffic. Possible options: ■ ITU14: ITU operation with 14 bit Point Code ■ ITU16: ITU operation with 16 bit Point Code ■ ITU24: ITU operation with 24 bit Point Code ■ ANSI: ANSI operation with 24 bit Point Code Default value: ITU14
Traffic Mode	STRING	Specifies the traffic mode in which SSUs operate. Possible options: ■ Loadshare (LS): SSU shares traffic distribution with any other currently active SSUs. ■ Broadcast (BC): SSU receives the same messages as any other currently active SSUs ■ Override (OR): SSU takes over all traffic in Service Broker (that is, primary/backup operation) overriding any currently active SSUs in Service Broker Default value: Loadshare (LS)

Table 2–6 (Cont.) Local Systems Fields

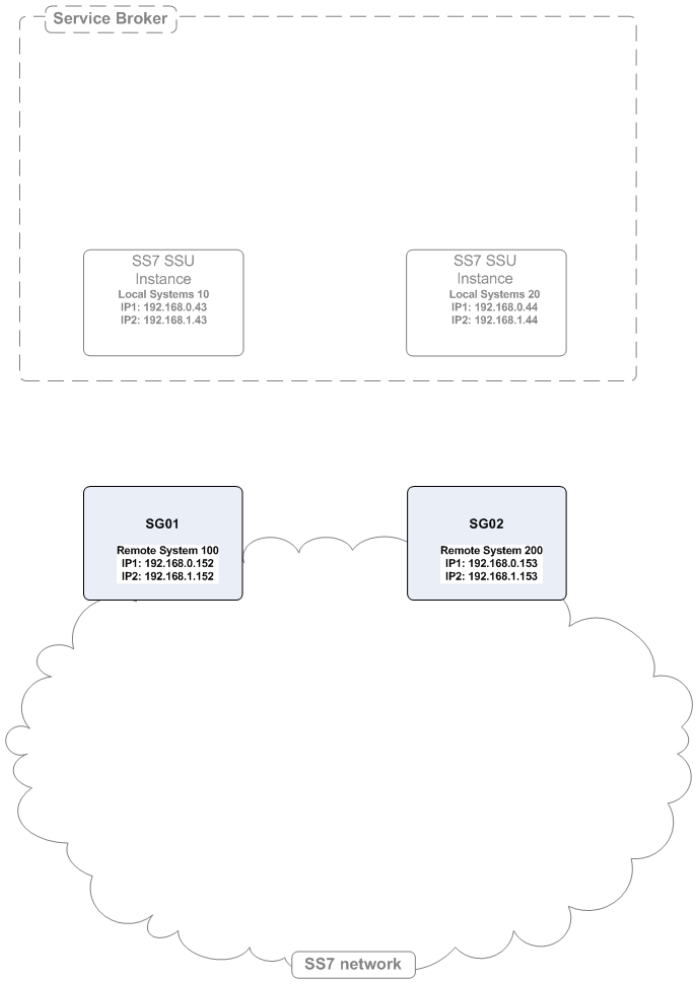
Name	Type	Description
IP Address1	STRING	Specifies an SSU IP. The IP address must have the following format: n.n.n.n. Default value: 0.0.0.0
IP Address2	STRING	Specifies an alternative SSU IP address. This address is used when the address defined in the IP Address1 parameter is unreachable. The IP address must have the following format: n.n.n.n.

Configuring Remote Systems

The Remote Systems subtab enables you to configure other M3UA network entities to which the SSU instance is directly connected.

Figure 2–2 shows an example of configuration of the remote systems components.

Figure 2–2 Configuration Example: M3UA Remote Systems



The Remote Systems subtab contains a table in which each row represents a single entity that acts as a remote system. When defining a remote system, you need to specify the fields described in Table 2–7.

Table 2–7 Remote Systems Fields

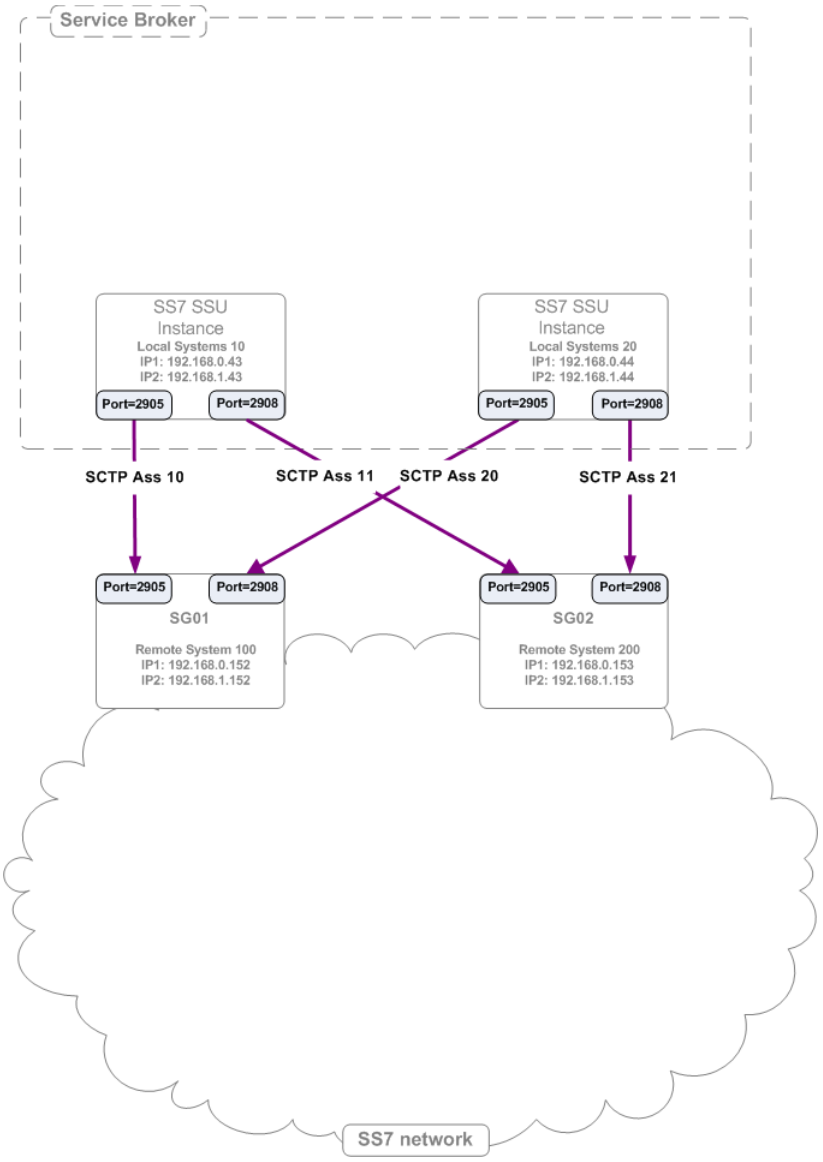
Name	Type	Description
Name	STRING	Specifies a unique name for the Remote System
Type	STRING	Specifies the network entity type. The only available option is SG which stands for Signaling Gateway.
IP Address 1	STRING	Specifies a network entity IP address. The IP address must have the following format: n.n.n.n. Default value: 0.0.0.0.
IP Address 2	STRING	Specifies a network entity alternative IP address. This address is used when the address defined in the IP Address 1 parameter is unreachable. The IP address must have the following format: n.n.n.n.

Network Mapping

The Network Mapping subtab enables you to define SCTP associations that connect a local system (an SSU instance) to remote systems.

Figure 2–3 shows an example of configuration of SCTP associations.

Figure 2–3 Configuration Example: M3UA SCTP Associations



The SCTP Associations subtab contains a table in which each row represents a single association. When defining an SCTP association, you need to specify the fields described in Table 2–8.

Table 2–8 SCTP Associations Fields

Name	Type	Description
Name	STRING	Specifies a descriptive name for the SCTP association
Side	STRING	Specifies the mode in which the local side operates. Possible options: ■ Client ■ Server Default value: Client. Setting this parameter requires coordination with the application on the remote side.

Table 2–8 (Cont.) SCTP Associations Fields

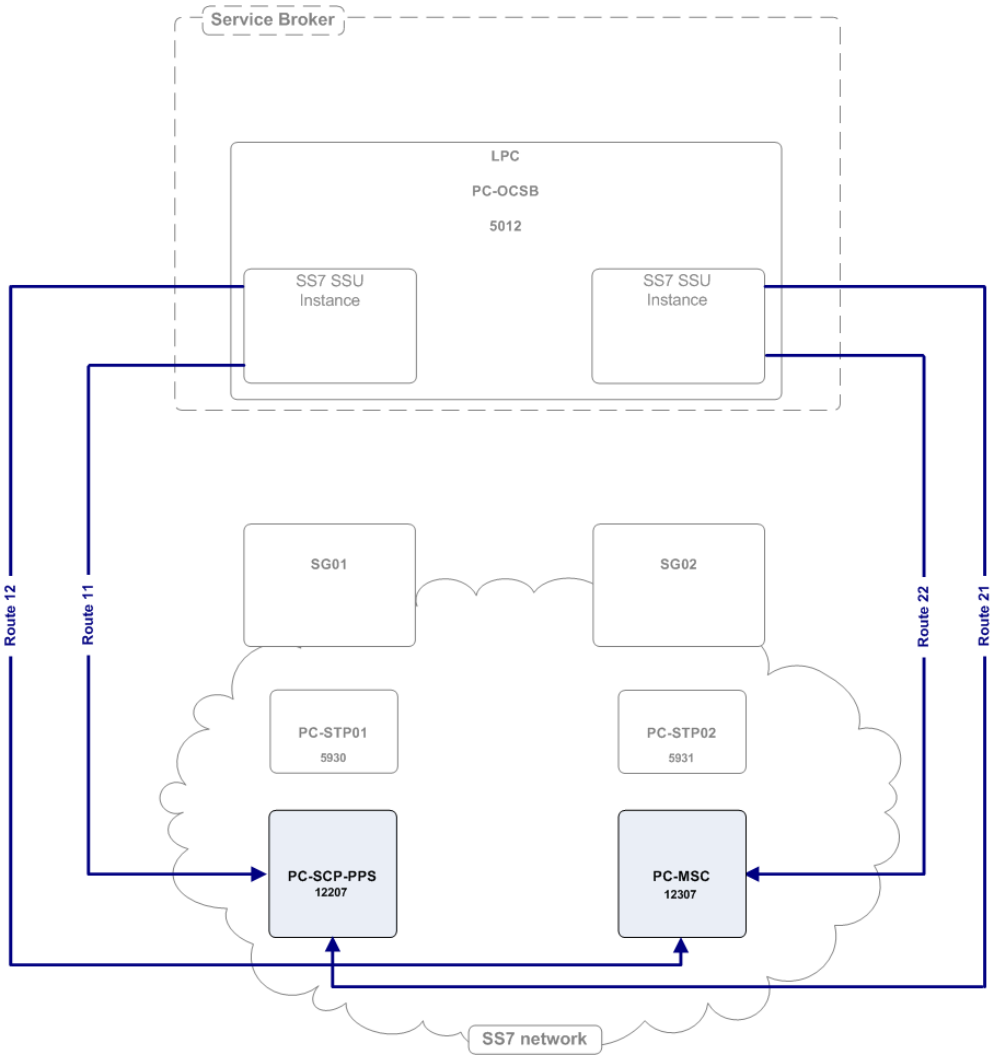
Name	Type	Description
Type	STRING	Specifies the SIGTRAN mode. Set this parameter to M3UA.
Local Port	INT	Specifies an SCTP port on the local system side.
Remote Side	STRING	Specifies an entity on the association's network side. Select one of the remote systems that you have previously defined on the Remote Systems subtab in the Connectivity section. See "Configuring Remote Systems" for more information about configuring remote systems.
Remote Port	INT	Specifies an SCTP port on the remote system side

Network Routing

The Network Routing subtab enables you to configure routes to entities in an SS7 network.

Figure 2–4 shows an example of configuration of M3UA routes.

Figure 2–4 Configuration Example: M3UA Routes



The M3UA Routes subtab contains a table in which each row represents a route. When defining a route, you need to specify the fields described in Table 2–9.

Table 2–9 M3UA Routes Fields

Name	Type	Description
Name	STRING	Specifies a descriptive name for the route
Remote Point Code	INT (11)	Specifies an RPC that is available on the far end of the route. You can select one of the RPCs that you have previously defined on the Point Codes subtab in the Network Mapping section.
Primary Remote SIGTRAN System	STRING	Specifies the remote SIGTRAN system through which the SSU instance routes messages to the remote entity. Most likely, this is a Signaling Gateway.
Secondary Remote SIGTRAN System	STRING	Specifies an alternative SIGTRAN system through which the SSU instance routes messages to the remote entity

SCCP

The SCCP tab enables you to configure SCCP addresses for:

- Service Broker modules
- Remote entities in an SS7 network.

To access the SCCP tab:

- In the SS7 SSU SIGTRAN configuration pane, click the SCCP tab.

  The SCCP configuration screen contains the subtabs described in Table 2–10.

Table 2–10 SCCP Section Subtabs

Subtab	Description
General	Enables you to specify parameters, which are common for all SCCP addresses. See "General" for more information.
Local SSNs	Enables you to assign subsystem numbers for Service Broker module instances. See "Local SSNs" for more information.
Local GTs	Enables you to configure Global Title addresses for Service Broker module instances. See "Local GTs" for more information.
Remote PC and SSN Addresses	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a point code and a subsystem number. See "Remote PC and SSN Addresses" for more information.
Remote Fixed GTs	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a fixed Global Title. See "Remote Fixed GTs" for more information.
Remote Dynamic GTs	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a dynamic Global Title. See "Remote Dynamic GTs" for more information.
Global Title Routing	Enables you to configure addresses of network entities that perform Global Title Translation. See "Global Title Routing" for more information.

General

The General subtab enables you to specify parameters, which are common for all SCCP addresses. Table 2–11 describes the parameter on the General subtab that you need to define.

Table 2–11 General Parameter

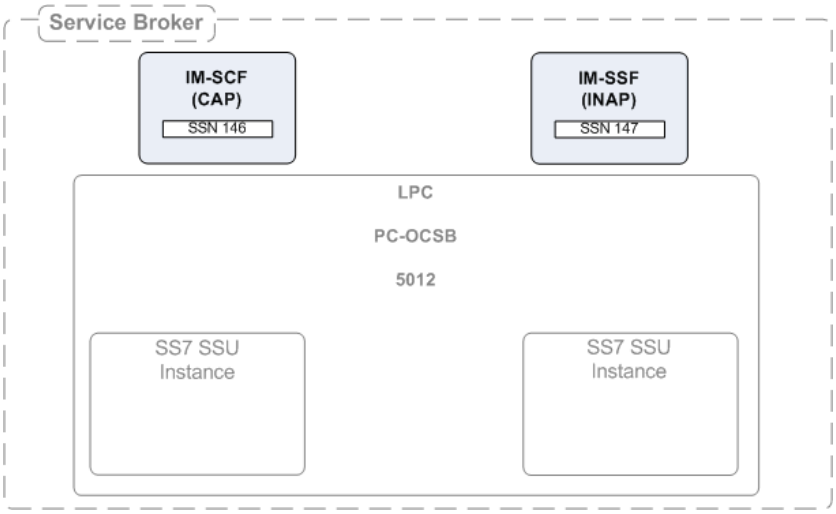
Name	Type	Description
Local Network Indicator	STRING	Specifies the network type of an SSU address, which is common for all SSU local SCCP addresses. Possible values: ■ International Network ■ International Network Extension ■ National Network ■ National Network Extension Default value: International Network The Local Network Indicator parameter of the M3UA stack is set to the same value as this parameter. However, because International Network Extension and National Network Extension are not supported in the M3UA stack, these two parameters are translated as follows in M3UA: ■ International Network Extension is translated to International Network ■ National Network Extension is translated to National Network
Remove Calling Party Point Code upon GT Routing	BOOL	Specifies whether or not the local SSU point code is to be added to the calling party address, when routing is done with a Global Title. Possible values: ■ True: the local point code is not added to the calling party address ■ False: the local point code is added to the calling party address
Remove Called Party Point Code upon GT Routing	BOOL	Specifies whether or not the remote point code is to be removed from the called party address, when routing is done with a Global Title. Possible values: ■ True: the remote point code is not added to the called party address ■ False: the remote point code is added to the called party address

Local SSNs

The Local SSNs subtab enables you to assign Subsystem Numbers (SSNs) for Service Broker module instances. An SSU routes incoming messages to local subsystems based on these SSNs.

Figure 2–5 shows an example of configuration of local SSNs.

Figure 2–5 Configuration Example: Local SSNs



The Local SSNs subtab contains a table in which each row represents a single Service Broker subsystem. When configuring an SSN, you need to specify the fields described in Table 2–12.

Table 2–12 Local SSNs Fields

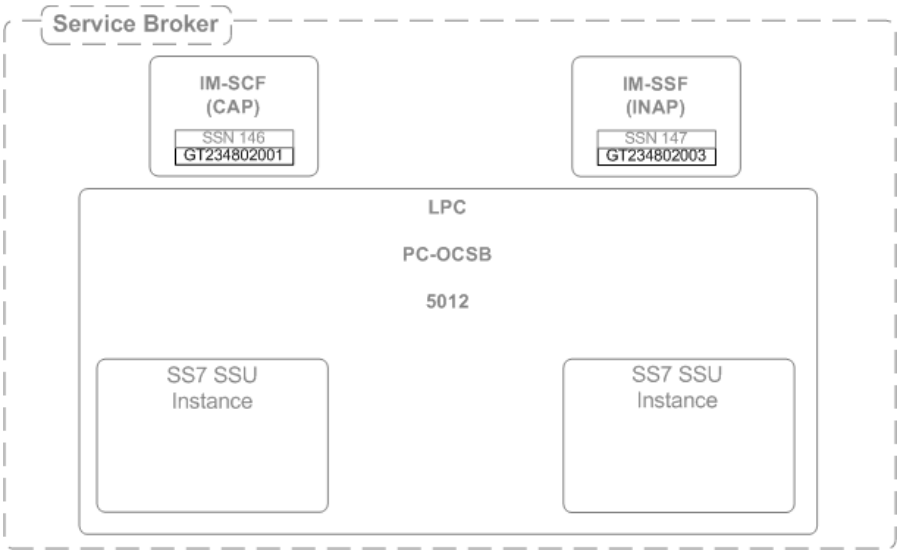
Name	Type	Description
Name	STRING	Specifies the subsystem name
SSN	INT	Specifies the subsystem number. Default value: 0.
Description	STRING	Specifies a subsystem description
Alias	STRING	Specifies an alias name given to a Service Broker subsystem. Applications that use Service Broker to connect to the SS7 network, use this alias to refer the specific subsystem.

Local GTs

The Local GTs subtab enables you to configure Global Title addresses for Service Broker module instances.

Figure 2–6 shows an example of configuration of local GTs.

Figure 2–6 Configuration Example: Local GT



The Local GTs subtab contains a table in which each row represents a single address. When defining an address, you need to specify the fields described in Table 2–13.

Table 2–13 Local GTs Fields

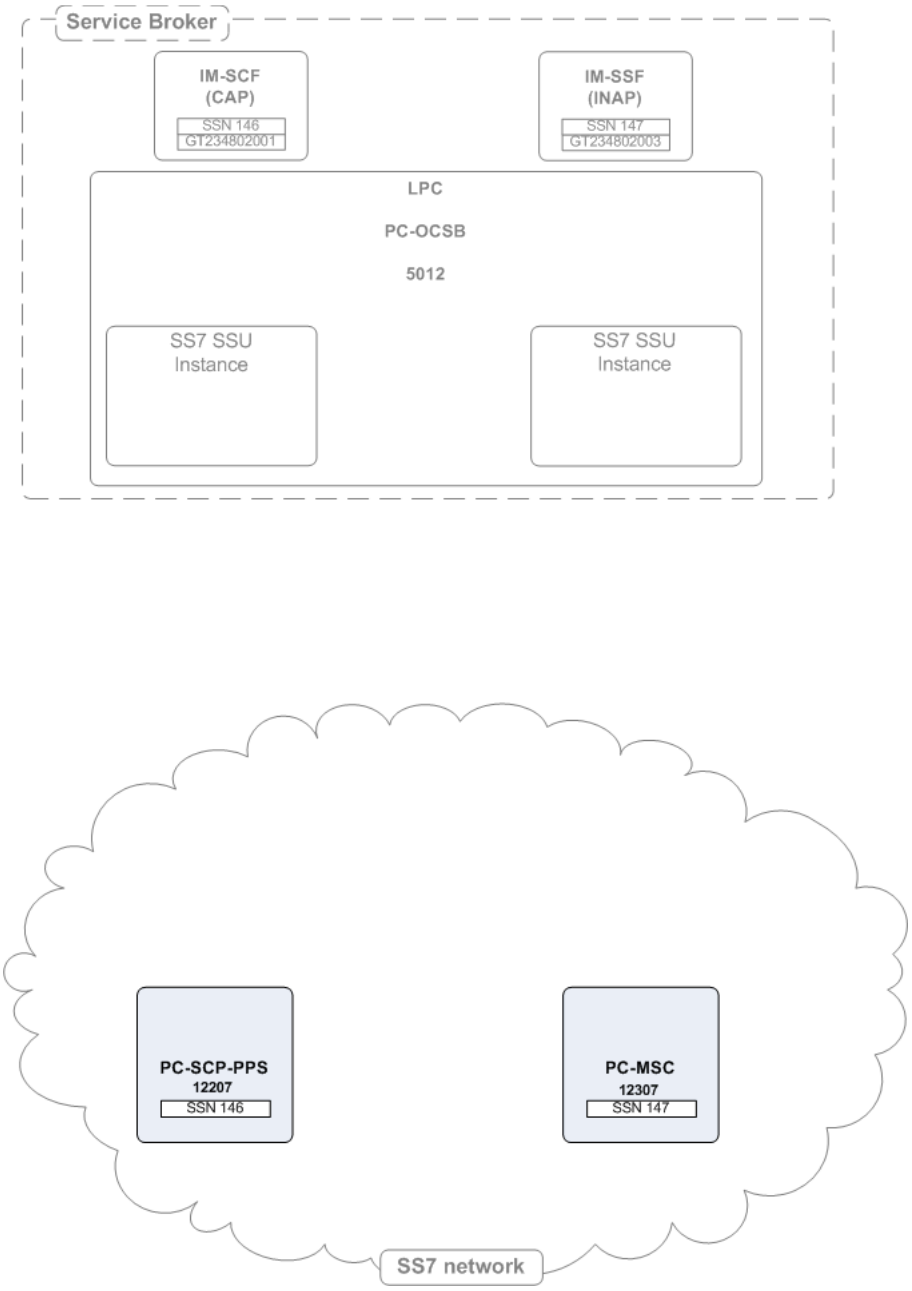
Name	Type	Description
Name	STRING	Specifies a unique name
Description	STRING	Specifies a description for the Service Broker GT address.
GT Address	STRING	Specifies the Global Title Address part of the SCCP address
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT.
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to a Service Broker subsystem. Applications that use Service Broker to connect to the SS7 network use this alias to refer the specific GT address.

Remote PC and SSN Addresses

The Remote PC and SSN Addresses subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a point code and a subsystem number.

Figure 2–7 shows an example of configuration of a remote point code and an SSN.

Figure 2–7 Configuration Example: Remote PC and SSN



The Remote PC and SSN Addresses subtab contains a table in which each row represents a single SS7 network entity. When configuring a network entity, you need to specify the fields described in Table 2–14.

Table 2–14 Remote PC and SSN Fields

Name	Type	Description
Name	STRING	Specifies a unique name

Table 2–14 (Cont.) Remote PC and SSN Fields

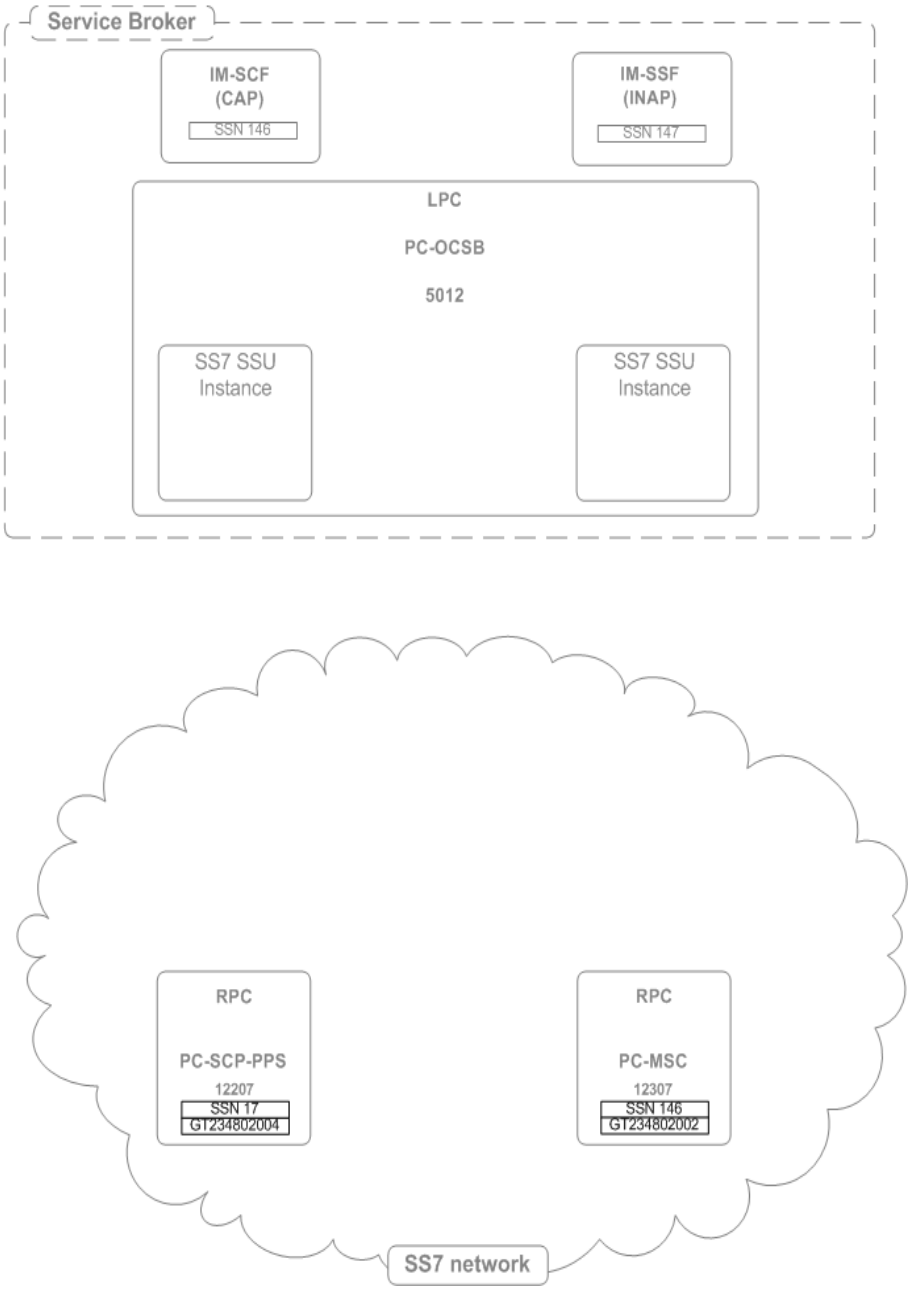
Name	Type	Description
Network Indicator	STRING	Specifies the network type. Possible values: ■ International Network ■ National Network Default value: International Network
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function.
Point Code	INT	Specifies the point code part of the SCCP address.
Description	STRING	Specifies a description for the remote SS7 network entity.
Alias	STRING	Specifies an alias name given to a remote network entity. Applications that use Service Broker to connect to the SS7 network use this alias to refer the specific network entity.

Remote Fixed GTs

The Remote Fixed GTs subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a fixed Global Title.

Figure 2–8 shows an example of configuration of remote fixed GTs.

Figure 2–8 Configuration Example: Remote Fixed GTs



The Remote Fixed GTs subtab contains a table in which each row represents a single SS7 network entity. When configuring a network entity, you need to specify the fields described in Table 2–15.

Table 2–15 Remote Fixed GTs Fields

Name	Type	Description
Name	STRING	Specifies a unique name

Table 2–15 (Cont.) Remote Fixed GTs Fields

Name	Type	Description
Network Indicator	STRING	Specifies the network type. Possible options: ■ International Network ■ National Network Default option: International Network
Description	STRING	Specifies a description for the network entity and its address
GT Address	STRING	Specifies the Global Title Address part of the SCCP address
Point Code	INT	Optional: specifies the point code part of the SCCP address. When specified, the SSU routes messages to the specified point code, including a GT address.
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT.
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to a remote network entity. Applications that use Service Broker to connect to the SS7 network use this alias to refer the specific network entity.

Remote Dynamic GTs

The Remote Dynamic GTs subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a dynamic Global Title.

The Remote Dynamic GTs subtab contains a table in which each row represents a single SCCP address. When configuring an SCCP address, you need to specify the fields described in Table 2–16.

Table 2–16 Remote Dynamic GTs Fields

Name	Type	Description
Name	STRING	Specifies a unique name
Network Indicator	STRING	Specifies the network type. The following options are available: ■ International Network ■ National Network Default value: International Network
Description	STRING	Specifies a description for the dynamic GT address
Point Code	INT	Optional: specifies the point code part of the SCCP address. When specified, the SSU routes messages to the specified point code, including a GT address.

Table 2–16 (Cont.) Remote Dynamic GTs Fields

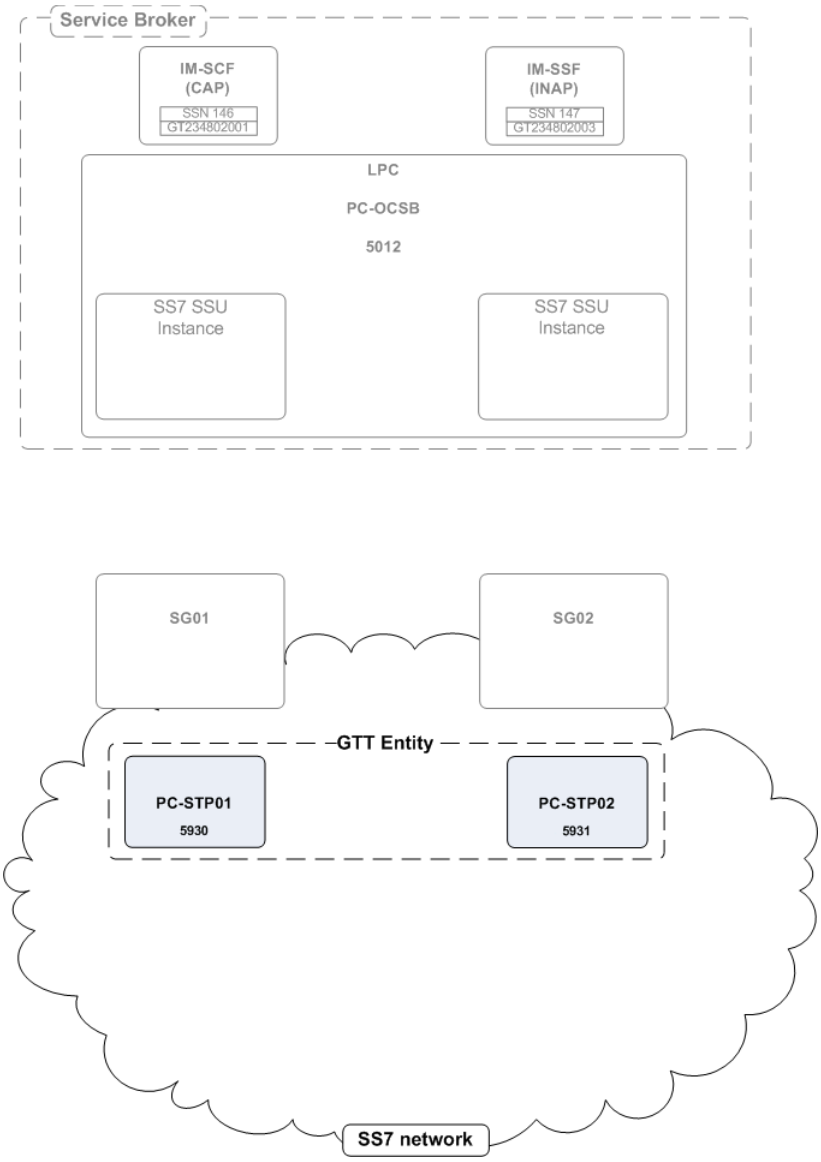
Name	Type	Description
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT.
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to an SCCP address. Applications that use Service Broker to connect to the SS7 network use this alias when they wish route messages using this address.

Global Title Routing

The Global Title Routing subtab enables you to configure addresses of network entities that perform Global Title Translation. Typically these point codes are Signal Transfer Points (STPs).

Figure 2–9 shows an example of configuration of point codes.

Figure 2–9 Configuration Example: Global Title Routing



The Global Title Routing subtab contains a table in which each row represents a point code that performs GTT. When defining a point code that performs GTT, you need to specify the fields described in Table 2–17.

Table 2–17 Global Title Routing Parameters

Name	Type	Description
Primary GTT Point Code	INT	Specifies a primary remote point code that performs GTT.
Secondary GTT Point Code	INT	Specifies an alternative remote point code that performs GTT.

Table 2–17 (Cont.) Global Title Routing Parameters

Name	Type	Description
Operation Mode	STRING	Specifies the mode in which the primary and secondary remote point codes operate. The following options are available: ■ LOAD_SHARING: the SSU sends messages to both primary and secondary point codes in a load sharing mode. ■ PRIMARY_SECONDARY: the SSU sends messages to the primary point code. If the primary point code is not available, the SSU routes messages to the secondary point code. Default value: PRIMARY_SECONDARY

Routing

The Routing tab enables you to define an IM to which the SS7 SSU routes an incoming session by specifying a set of parameters known as incoming routing rule. For each incoming routing rule, you need to configure the following parameters:

- IM to which the SS7 SSU routes an incoming session
- Criteria that an incoming session must meet to be routed to this IM
- Priority in which the SS7 SSU checks incoming routing rules to evaluate whether an incoming session fits the criteria defined in a rule. The SS7 SSU applies the first found rule which criteria are met by an incoming session.

  For example, if you created multiple rules for the same IM, SS7 SSU begins with the rule that has the highest priority. If an incoming session fits the criteria defined in this rule, the SS7 SSU applies the rule and do not check the rest of the rules. Otherwise, the SS7 SSU checks whether an incoming session fits the criteria of a rule with a lower priority. The SS7 SSU performs this check until the SS7 SSU finds a rule whose criteria are met by an incoming session.

You can define incoming routing rules using the Routing tab. The process of defining an incoming routing rule consists of the following steps:

1. You create a rule and define its name, priority, and an IM for which you create this rule. You perform these actions using the Incoming Routing Rules subtab.
2. You define criteria for each rule that you created on step 1.

Accessing the Routing Tab

The Routing tab enables you to define rules for routing incoming sessions to IMs.

To access the Routing tab:

1. In the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Select **SSU SS7 SIGTRAN**.
4. Click the **Routing** tab.

   The Routing tab contains the subtabs described in Table 2–18.

Table 2–18 Routing Subtabs

Subtab	Description
Incoming Routing Rules	Enables you to define a name, priority, and an IM for which you create a rule. See "Configuring Incoming Routing Rules Parameters" for more information.
Incoming Routing Criteria	Enables you to define criteria for each routing rule created on the Incoming Routing Rules subtab. See "Configuring Incoming Routing Criteria Parameters" for more information.

Configuring Incoming Routing Rules Parameters

The Incoming Routing Rules subtab enables you to define a name, priority, and an IM for which you create a rule. The Incoming Routing Rules subtab contains a table in which each row represents an individual rule.

When you define a rule, you need to specify the fields defined in Table 2–19.

Table 2–19 Incoming Routing Rule Fields

Name	Type	Description
Name	STRING	Specifies a unique rule name
Priority	INT	Specifies an order in which the SS7 SSU checks routing rules to evaluate if an incoming session fits rule's criteria. The SS7 SSU applies the first found rule which criteria are met by an incoming session. The lower the number, the higher the priority. For example, if you created two rules and set Priority of one rule to" 1" and set Priority of another rule to "2", the SS7 SSU checks the rule with Priority set to "1" first. You can define an incoming routing rule that the SS7 SSU apply if no other rule can be applied by setting the Priority parameter of this rule to the highest number (that is, the number with the lowest priority). There is no need to specify incoming routing criteria for such a rule.

Table 2–19 (Cont.) Incoming Routing Rule Fields

Name	Type	Description
Module Instance	STRING	Specifies a URI of an IM to which the SS7 SSU routes an incoming session. The URI has the following format: *IM-instance-name.IM-type@domain-id* ■ *IM-instance-name*: The IM instance name that you specified when you added this IM in the IM Management Configuration screen. ■ *IM-type*: The type of the IM instance ■ *domain-id*: The name of a Processing Domain or a Processing Domain Group where the relevant IM is deployed. See "Setting Up the Service Broker Domain Name" in the *Oracle Communications Service Broker System Administrator's Guide* for more information on setting up a domain name. To set a Processing Domain, you must specify the name you configured for the domain during its creation. See "Setting a Service Broker Domain Name" in Oracle *Communications Service Broker Processing Domain Configuration Guide* for more information. To set a Processing Domain Group, you must specify the group name. See Managing Processing Domain Groups in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information about Processing Domain Groups. Example: `imscfcap4_instance.IMSCFCAP4@processing-domain-1`

Configuring Incoming Routing Criteria Parameters

The Incoming Routing Criteria subtab enables you to define criteria for rules that you created on the Incoming Routing Rules subtab. The Incoming Routing Criteria contains a table in which each row represents a routing rule.

When you define criteria, you need to specify the fields defined in Table 2–20.

Table 2–20 Incoming Routing Criteria Fields

Name	Type	Description
Parent	STRING	Specifies the name of a rule for which you want to define criteria. This list contains names of the rules that you created on the Incoming Routing Rules subtab.
Name	STRING	Specifies a unique rule name
Session Key	STRING	Specifies a parameter inside an SCCP message based on which the SS7 SSU performs routing. The SS7 SSU routes incoming messages to a specified module instance, if the value of this parameter matches the Value field. Possible values: ■ DEST_ADDRESS_ALIAS ■ SOURCE_ADDRESS_ALIAS ■ APPLICATION_CONTEXT ■ SERVICE_KEY ■ OPCODE

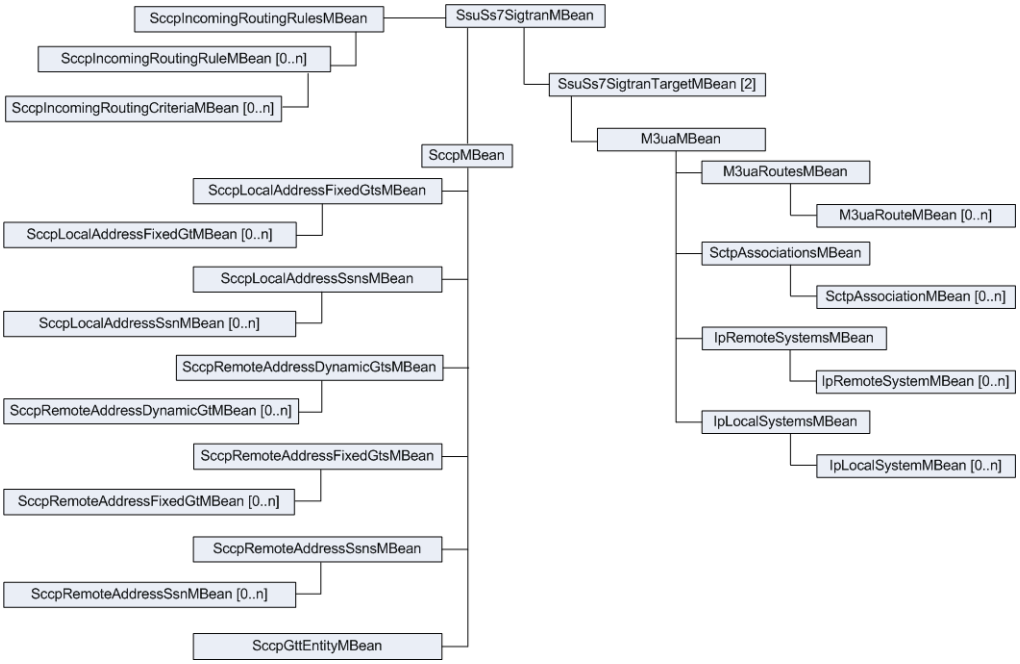
Table 2–20 (Cont.) Incoming Routing Criteria Fields

Name	Type	Description
Value	STRING	Specifies a value that the Session Key parameter of an SCCP message must match, in order for the rule specified in Parent to apply. You can define one of the following in the Value parameter: ■ Single value ■ Range of dash-separated values ■ Comma-separated values

Configuring SS7 SSU for SIGTRAN with Java MBeans

Service Broker provides a set of MBeans that expose attributes and operations for configuring SSU SS7 for SIGTRAN through JMX. Figure 2–10 shows the hierarchy of MBeans that you use for configuring SSU SS7 for SIGTRAN. Because a Service Broker deployment includes two SSU instances, you need to configure an SsuTargetMBean for each SSU instance separately.

Figure 2–10 SSU SS7 for SIGTRAN Configuration MBean Hierarchy



The following sections provide reference information for the SS7 SSU configuration MBeans.

Note: MBeans described in this guide might include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

SsuSs7SigtranMBean

Ss7SsuSigtranMBean is a root MBean for the SS7 SSU for SIGTRAN configuration.

Factory Method

Created automatically

Attributes

- Vendor
- Standard
- Ss7StackIp
- Ss7StackPort

For more information on these attributes, see Table 2–2.

Operations

ObjectName getSccp()

Gets a reference to the instance of SccpMBean

ObjectName[] getSsuSs7SigtranTargets()

Gets an array of references to instances of SsuSs7SigtranTargetMBean

ObjectName getSccpIncomingRoutingRules()

Gets a reference to the instance of SccpIncomingRoutingRulesMBean

ObjectName createSsuSs7SigtranTarget()

Creates a new instance of SsuSs7SigtranTargetMBean

void destroySsuSs7SigtranTarget()

Destroys an existing instance of SsuSs7SigtranTargetMBean

ObjectName lookupSsuSs7SigtranTarget()

Returns the SsuSs7SigtranTargetMBean of the SSU running on the specified target server

SsuSs7SigtranTargetMBean

SsuSs7SigtranTargetMBean enables you to access the configuration MBeans of a specific SSU

Factory Method

Ss7SsuSigtran.createSsuSs7SigtranTarget()

Attributes

None

Operations

ObjectName getM3ua()

Returns a reference to the instance of M3uaMBean

M3uaMBean

M3uaMBean enables you to access the configuration MBeans of a specific SS7 SSU instance.

Factory Method

Created automatically

Attributes

LocalPointCode

For more information on these attributes, see Table 2–4.

Operations

ObjectName getIpLocalSystems()

Returns a reference to the instance of IpLocalSystemsMBean

ObjectName getIpRemoteSystems()

Returns a reference to the instance of IpRemoteSystemsMBean

ObjectName getM3uaRoutes()

Returns a reference to the instance of M3uaRoutesMBean

ObjectName getSctpAssociations()

Returns a reference to the instance of SctpAssociationsMBean

IpLocalSystemsMBean

IpLocalSystemsMBean is a root MBean for individual instances of IpLocalSystemMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getIpLocalSystems()

Gets an array of references to instances of IpLocalSystemMBean

ObjectName createIpLocalSystem()

Creates a new instance of IpLocalSystemMBean

void destroyIpLocalSystem()

Destroys an existing instance of IpLocalSystemMBean

ObjectName lookupIpLocalSystem()

Returns a reference to the specified instance of IpLocalSystemMBean

IpLocalSystemMBean

IpLocalSystemMBean enables you to configure an SS7 SSU instance as a local SIGTRAN system.

Factory Method

IpLocalSystems.createIpLocalSystem()

Attributes

- Name
- RoutingContext
- SS7Mode
- TrafficMode
- FirstIp
- SecondIp

For more information on these attributes, see Table 2–6.

Operations

None

IpRemoteSystemsMBean

IpRemoteSystemsMBean is a root MBean for individual instances of IpRemoteSystemMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getIpRemoteSystems()

Gets an array of references to instances of IpRemoteSystemMBean

ObjectName createIpRemoteSystem()

Creates a new instance of IpRemoteSystemMBean

void destroyIpRemoteSystem()

Destroys an existing instance of IpRemoteSystemMBean

ObjectName lookupIpRemoteSystem()

Returns a reference to the specified instance of IpRemoteSystemMBean

IpRemoteSystemMBean

IpRemoteSystemMBean enables you to configure an M3UA network entity to which the SSU instance is directly connected.

Factory Method

IpRemoteSystems.createIpRemoteSystem()

Attributes

- Name
- Type
- FirstIp
- SecondIp

For more information on these attributes, see Table 2–7.

Operations

None

SctpAssociationsMBean

SctpAssociationsMBean is a root MBean for individual instances of SctpAssociationMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSctpAssociations()

Gets an array of references to instances of SctpAssociationMBean

ObjectName createSctpAssociation()

Creates a new instance of SctpAssociationMBean

void destroySctpAssociation()

Destroys an existing instance of SctpAssociationMBean

ObjectName lookupSctpAssociation()

Returns a reference to the specified instance of SctpAssociationMBean

SctpAssociationMBean

SctpAssociationMBean enables you to configure a channel that connects an SSU instance to another SIGTRAN network entity.

Factory Method

SctpAssociations.createSctpAssociation()

Attributes

- Name
- Side
- Type
- LocalPort
- RemoteSide
- RemotePort

For more information on these attributes, see Table 2–8.

Operations

None

M3uaRoutesMBean

M3uaRoutesMBean is a root MBean for individual instances of M3uaRouteMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getM3uaRoutes()

Gets an array of references to instances of M3uaRouteMBean

ObjectName createM3uaRoute()

Creates a new instance of M3uaRouteMBean

void destroyM3uaRoute()

Destroys an existing instance of M3uaRouteMBean

ObjectName lookupM3uaRoute()

Returns a reference to the specified instance of M3uaRouteMBean

M3uaRouteMBean

M3uaRouteMBean enables you to configure how SS7 SSU instances can access SS7 network entities.

Factory Method

M3uaRoutes.createM3uaRoute()

Attributes

- Name
- RemotePointCode
- PrimaryRemoteSigtranSystem
- SecondaryRemoteSigtranSystem

For more information on these attributes, see Table 2–9.

Operations

None

SccpIncomingRoutingRulesMBean

SccpIncomingRoutingRulesMBean enables you to define how the SS7 SSU routes incoming SS7 messages to Service Broker IMs.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpIncomingRoutingRules()

Gets an array of references to instances of SccpIncomingRoutingRuleMBean

ObjectName createSccpIncomingRoutingRule()

Creates a new instance of SccpIncomingRoutingRuleMBean

void destroySccpIncomingRoutingRule()

Destroys an existing instance of SccpIncomingRoutingRuleMBean

ObjectName lookupSccpIncomingRoutingRule()

Returns a reference to the SccpIncomingRoutingRuleMBean

SccpIncomingRoutingRuleMBean

SccpIncomingRoutingRuleMBean represents an individual incoming routing rule.

Factory Method

SccpIncomingRoutingRules.createSccpIncomingRoutingRule()

Attributes

- Name
- Priority
- ModuleInstance

For more information on these attributes, see Table 2–19.

Operations

ObjectName[] getSccpIncomingRoutingCriteria()

Gets an array of references to instances of SccpIncomingRoutingCriteriaMBean

ObjectName createSccpIncomingRoutingCriteria()

Creates a new instance of SccpIncomingRoutingCriteriaMBean

void destroySccpIncomingRoutingCriteria()

Destroys an existing instance of SccpIncomingRoutingCriteriaMBean

ObjectName lookupSccpIncomingRoutingCriteria()

Returns a reference to the SccpIncomingRoutingCriteriaMBean

SccpIncomingRoutingCriteriaMBean

SccpIncomingRoutingCriteriaMBean represents criteria for an individual incoming routing rule.

Factory Method

SccpIncomingRoutingRules.createSccpIncomingRoutingCriteria()

Attributes

- Parent
- Name
- SessionKey
- Value

For more information on these attributes, see Table 2–20.

Operations

None

SccpMBean

SccpMBean enables you to configure SCCP addresses: subsystems and global titling.

Factory Method

Created automatically

Attributes

- LocalNetworkIndicator
- RemoveCallingPartyPointCodeUponGtRouting
- RemoveCalledPartyPointCodeUponGtRouting

For more information, see Table 2–11.

Operations

ObjectName getSccpGttEntity()

Gets a reference to the instance of SccpGttEntityMBean

ObjectName[] getSccpLocalAddressFixedGts()

Gets an array of references to instances of SccpLocalAddressFixedGtMBean

ObjectName[] getSccpLocalAddressSsns()

Gets an array of references to instances of SccpLocalAddressSsnMBean

ObjectName[] getSccpRemoteAddressDynamicGts()

Gets an array of references to instances of SccpRemoteAddressDynamicGtMBean

ObjectName[] getSccpRemoteAddressFixedGts()

Gets an array of references to instances of SccpRemoteAddressFixedGtMBean

ObjectName[] getSccpRemoteAddressSsns()

Gets an array of references to instances of SccpRemoteAddressSsnMBean

SccpLocalAddressFixedGtsMBean

SccpLocalAddressFixedGtsMBean is a root MBean for individual instances of SccpLocalAddressFixedGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpLocalAddressFixedGts()

Gets an array of references to instances of SccpLocalAddressFixedGtMBean

ObjectName createSccpLocalAddressFixedGt()

Creates a new instance of SccpLocalAddressFixedGtMBean

void destroySccpLocalAddressFixedGt()

Destroys an existing instance of SccpLocalAddressFixedGtMBean

ObjectName lookupSccpLocalAddressFixedGt()

Returns a reference to the specified instance of SccpLocalAddressFixedGtMBean

SccpLocalAddressFixedGtMBean

SccpLocalAddressFixedGtMBean enables you to configure a Global Title address for a Service Broker module instance.

Factory Method

SccpLocalAddressFixedGts.createSccpLocalAddressFixedGt()

Attributes

- Name
- Description
- GtAddress
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 2–13.

Operations

None

SccpLocalAddressSsnsMBean

SccpLocalAddressSsnsMBean is a root MBean for individual instances of SccpLocalAddressSsnMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpLocalAddressSsns()

Gets an array of references to instances of SccpLocalAddressSsnMBean

ObjectName createSccpLocalAddressSsn()

Creates a new instance of SccpLocalAddressSsn

void destroySccpLocalAddressSsn()

Destroys an existing instance of SccpLocalAddressSsn

ObjectName lookupSccpLocalAddressSsn()

Returns a reference to the specified instance of SccpLocalAddressSsn

SccpLocalAddressSsnMBean

SccpLocalAddressSsnMBean enables you to assign a subsystem number to a Service Broker module instance.

Factory Method

SccpLocalAddressSsns.createSccpLocalAddressSsn()

Attributes

- Name
- Ssn
- Description
- Alias

For more information on these attributes, see Table 2–12.

Operations

None

SccpRemoteAddressDynamicGtsMBean

SccpRemoteAddressDynamicGtsMBean is a root MBean for individual instances of SccpRemoteAddressDynamicGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressDynamicGts()

Gets an array of references to instances of SccpRemoteAddressDynamicGtMBean

ObjectName createSccpRemoteAddressDynamicGt()

Creates a new instance of SccpRemoteAddressDynamicGtMBean

void destroySccpRemoteAddressDynamicGt()

Destroys an existing instance of SccpRemoteAddressDynamicGtMBean

ObjectName lookupSccpRemoteAddressDynamicGt()

Returns a reference to the specified instance of SccpRemoteAddressDynamicGtMBean

SccpRemoteAddressDynamicGtMBean

SccpRemoteAddressDynamicGTMBean enables you to configure an address of a remote entity in the SS7 network that can be reached using a dynamic Global Title.

Factory Method

SccpRemoteAddressDynamicGts.createSccpRemoteAddressDynamicGt()

Attributes

- Name
- NetworkIndicator
- Description
- PointCode
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 2–16.

Operations

None

SccpRemoteAddressFixedGtsMBean

SccpRemoteAddressFixedGtsMBean is a root MBean for individual instances of SccpRemoteAddressFixedGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressFixedGts()

Gets an array of references to instances of SccpRemoteAddressFixedGtMBean

ObjectName createSccpRemoteAddressFixedGt()

Creates a new instance of SccpRemoteAddressFixedGtMBean

void destroySccpRemoteAddressFixedGt()

Destroys an existing instance of SccpRemoteAddressFixedGtMBean

ObjectName lookupSccpRemoteAddressFixedGt()

Returns a reference to the specified instance of SccpRemoteAddressFixedGtMBean

SccpRemoteAddressFixedGtMBean

SccpRemoteAddressFixedGTMBean enables you to configure an address of a remote entity in the SS7 network that can be reached using a fixed Global Title.

Factory Method

SccpRemoteAddressFixedGts.createSccpRemoteAddressFixedGt()

Attributes

- Name
- NetworkIndicator
- Description
- GtAddress
- PointCode
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 2–15.

Operations

None

SccpRemoteAddressSsnsMBean

SccpRemoteAddressSsnsMBean is a root MBean for individual instances of SccpRemoteAddressSsnMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressSsns()

Gets an array of references to instances of SccpRemoteAddressSsnMBean

ObjectName createSccpRemoteAddressSsn()

Creates a new instance of SccpRemoteAddressSsnMBean

void destroySccpRemoteAddressSsn()

Destroys an existing instance of SccpRemoteAddressSsnMBean

ObjectName lookupSccpRemoteAddressSsn()

Returns a reference to the specified instance of SccpRemoteAddressSsnMBean

SccpRemoteAddressSsnMBean

SccpRemoteAddressSsnMBean enables you to configure an address of a remote entity in the SS7 network, that can be reached using a point code and a subsystem number.

Factory Method

SccpRemoteAddressSsns.createSccpRemoteAddressSsn()

Attributes

- Name
- NetworkIndicator
- Ssn
- PointCode
- Description
- Alias

For more information on these attributes, see Table 2–14.

Operations

None

SccpGttEntityMBean

SccpGttEntityMBean enables you to configure an address of a network entity that performs Global Title Translation.

Factory Method

Created automatically

Attributes

- PrimaryPointCode
- SecondaryPointCode
- OperationMode

For more information on these attributes, see Table 2–17.

Operations

None

3

Configuring the SS7 Signaling Server Unit for TDM

The following sections describe how to configure Service Broker SS7 SSUs using the Service Broker Administration Console and Java MBeans.

Configuring SS7 SSU TDM with the Administration Console

This section describes how to configure an SS7 SSU in a network in which SS7 traffic is carried over the traditional TDM. Because deployment of Service Broker involves configuration of two SSU instances, SS7 equipment connected to both instances must be configured as described in the following sections. The section provides graphical representations of deployment examples.

You can configure SS7 SSUs using the SSU SS7 TDM configuration pane.

To access the SSU SS7 TDM configuration pane:

1. In the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Select **SSU SS7 TDM**.

   The SSU SS7 configuration pane contains the subtabs described in Table 3–1.

> **Note:** You must configure the parameters exactly in the same order as they are presented in Table 3–1.

Table 3–1 SSU SS7 TDM Tabs

Tab	Description
SS7 SSU TDM	Enables you to assign a point code to local SSU instances and configure the MTP stack runtime options. See "SSU SS7 TDM" for more information.
MTP	Enables you to configure the MTP layers of the SS7 stack. See "MTP" for more information.
SCCP	Enables you to configure SCCP addresses: subsystems and global titling. See "SCCP" for more information.

Table 3–1 (Cont.) SSU SS7 TDM Tabs

Tab	Description
Routing	Enables you to define how the SS7 SSU routes incoming SS7 messages to internal Service Broker IMs. See "Routing" for more information.

SSU SS7 TDM

The SSU SS7 TDM tab enables you to assign a point code to a Service Broker SSU and configure the MTP stack runtime options.

To access the SSU SS7 TDM tab:

- In the SSU SS7 TDM configuration pane, click the **SSU SS7 TDM** tab.

  The SS7 SSU TDM configuration pane contains the parameters described in Table 3–2.

Table 3–2 SSU SS7 TDM Parameters

Name	Type	Description
Board Type	STRING	Specifies the board density. Possible values: ■ High: High density ■ Low: Low density Default value: High
Vendor	STRING	Specifies an MTP stack vendor. Possible values: ■ isigtran ■ dialogic
Standard	STRING	Specifies the standard that the MTP stack must use. Possible values: ■ ANSI ■ ETSI Default value: ETSI
MTP3RPO	STRING	Specifies the method of handling the Remote Processor Outage (RPO). Possible values: ■ Yes: Upon RPO, put the link in the Out of Service mode and select an alternative link. ■ No: Upon RPO, activate a timer first. Only if the failure remains by the time that the timer expires, move the link to the Out of Service mode and select an alternative link. The messages pulled up during time activation are discarded. Default value: Yes

Table 3–2 (Cont.) SSU SS7 TDM Parameters

Name	Type	Description
Routeset Test	STRING	Specifies whether or not the MTP RouteSetTest message must be sent when an RPC becomes unavailable. Possible values: ■ Yes – disable RouteSetTest ■ No – enable RouteSetTest Default value: Yes
SS7 Stack IP	INT	The IP address where the SS7 process is running. See "Starting and Stopping the SS7 Process" in *Service Broker System Administrator's Guide*.
SS7 Stack Port	INT	The port that the SS7 process is using to listen to messages from the SS7 SSU. This is the same port you specify to the SS7 process, in the command line, when you start it. See "Starting and Stopping the SS7 Process" in *Service Broker System Administrator's Guide*.

MTP

The MTP tab enables you to configure the MTP layers of the SS7 stack.

To access the MTP tab:

1. In the SS7 SSU TDM configuration pane, click the **MTP** tab.
2. In the **SSU Instance** list, select the Signaling Server that you want to configure.

   The SSU SS7 TDM configuration pane contains the tabs described in Table 3–3.

Table 3–3 MTP Subtabs

Subtab	Description
Local Point Code	Enables you to specify a point code for each SSU instance. For more information, see "Local Point Code".
Connectivity	Enables you to configure boards and PCM interfaces (E1/T1). For more information, see "Connectivity".
Network Mapping	Enables you to configure SS7 Links and Linksets that connect SSU to adjacent signaling points. For more information, see "Network Mapping".
Network Routing	Enables you to configure how an SSU accesses SS7 network entities. For more information, see "Network Routing".

Local Point Code

The Local Point Code subtab enables you to specify a point code of the SSU instance that you selected in the SSU Instance list, as described in Table 3–4.

Table 3–4 Point Code Field

Name	Type	Description
Local Point Code	INT	Specifies a local point code of the SSU instance that you selected in the SSU Instance list. A value of the parameter must be integer.

Connectivity

The Connectivity subtab enables you to configure boards and PCM interfaces (E1/T1).

Table 3–5 describes subtabs in the SSU SS7 Connectivity section.

Table 3–5 SS7 Connectivity Sections

Subtab	Description
Boards	Enables you to configure SS7 boards plugged into the SS7 SSU machine. For more information, see "Configuring SS7 Boards".
PCMs	Enables you to configure the PCMs that physically connect the SS7 SSU to the SS7 network. For more information, see "Configuring SS7 PCMs".

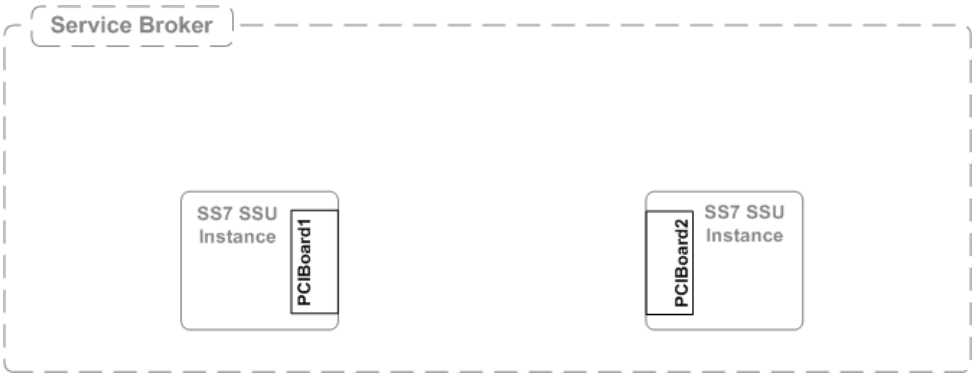
Configuring SS7 Boards

The Boards subtab enables you to configure SS7 boards plugged into machine chassis of the SSU instance.

An SS7 SSU instance can manage several SS7 PCI boards, depending on chassis and driver specifications.

Figure 3–1 shows an example of physical location of PCI boards.

Figure 3–1 Configuration Example: TDM Boards



The Boards subtab contains a table in which each row represents a single board. When defining a board, you need to specify the fields described in Table 3–6.

Table 3–6 SS7 Board Fields

Name	Type	Description
Name	STRING	Specifies a short name for the board.

Table 3–6 (Cont.) SS7 Board Fields

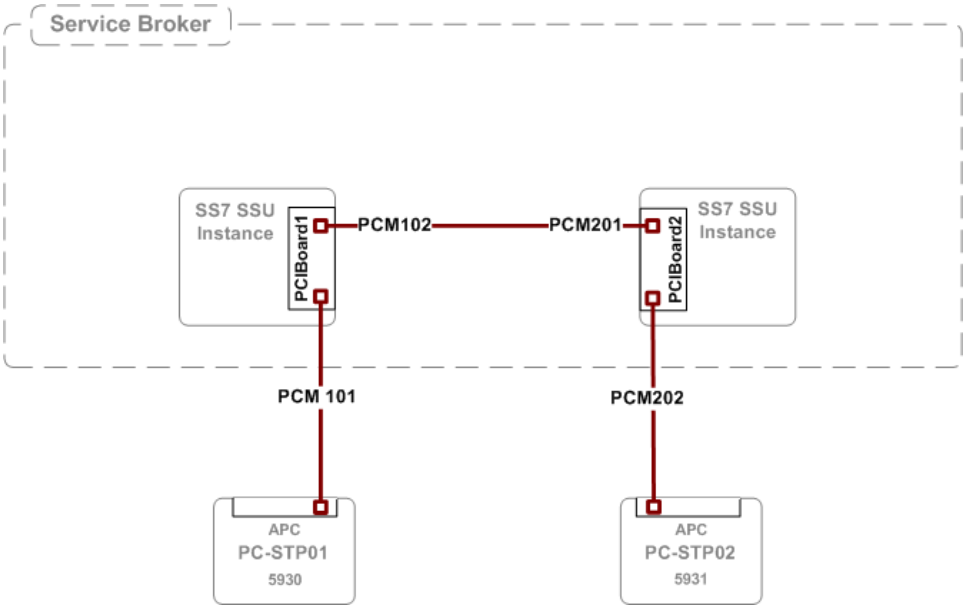
Name	Type	Description
Slot	INT	Specifies an SSU chassis slot number into which the board is plugged. Default value: 0
Clock	STRING	Specifies how the board is synchronized. Possible options: ■ Master: The board uses an external clock from one of its link interfaces to drive the bus clock for other boards on the bus. ■ Slave: The board uses the bus clocks, which must be generated by another board on the bus. ■ Internal: The board uses the onboard clock oscillator to drive the bus clock for other boards on the bus. Default value: Master

Configuring SS7 PCMs

The SS7 boards are standard PCI boards. The PCMs subtab enables you to define PCMs that physically connect an SSU instance to an SS7 network.

Figure 3–2 shows an example of a physical connection between SSU and an SS7 network.

Figure 3–2 Configuration Example: PCM



The PCMs subtab contains a table in which each row represents one PCM. When defining a PCM, you need to specify the fields described inTable 3–7.

Table 3–7 SS7 PCM Fields

Name	Type	Description
Name	STRING	Specifies a unique PCM name.

Table 3–7 (Cont.) SS7 PCM Fields

Name	Type	Description
Type	STRING	Specifies the type of the PCM hardware. Possible values: ■ DISABLED ■ E1-75ohm ■ E1-120ohm ■ T1, E1-75/120 ohm Default value: DISABLED
CRC	STRING	Specifies the CRC mode of operation. Possible values: ■ DISABLED ■ CRC4 ■ CRC4 compatibility mode ■ CRC6 enabled Default value: DISABLED
Code	STRING	Specifies the line encoding format. Possible values: ■ HDB3 (E1 only) ■ AMI with no zero code ■ AMI with zero code (T1 only) ■ B8ZS (T1 only) Default value: HDB3 (E1 only). **Note:** Code must match the Type parameter, for example, if the code/frame value is E1 only, then type should be E1.
Frame	STRING	Specifies the framing format. Possible values: ■ E1 double frame ■ E1 CRC4 multi-frame ■ D3/D4 (T1 only) ■ ESF (T1 only) Default value: E1 double frame. **Note:** Frame must match the Type parameter, for example, if the code/frame value is E1 only, then type should be E1.
Port	INT	Specifies a port number of the SS7 board into which the PCM is plugged. Possible values: ■ 0 ■ 1 ■ 2 ■ 3 Default value: 0

Network Mapping

The Network Mapping subtab enables you to configure SS7 Links and Linksets that connect SSU to adjacent signaling points.

The Network Mapping Configuration screen contains the subtabs described in Table 3–8.

Table 3–8 Network Mapping Section Subtabs

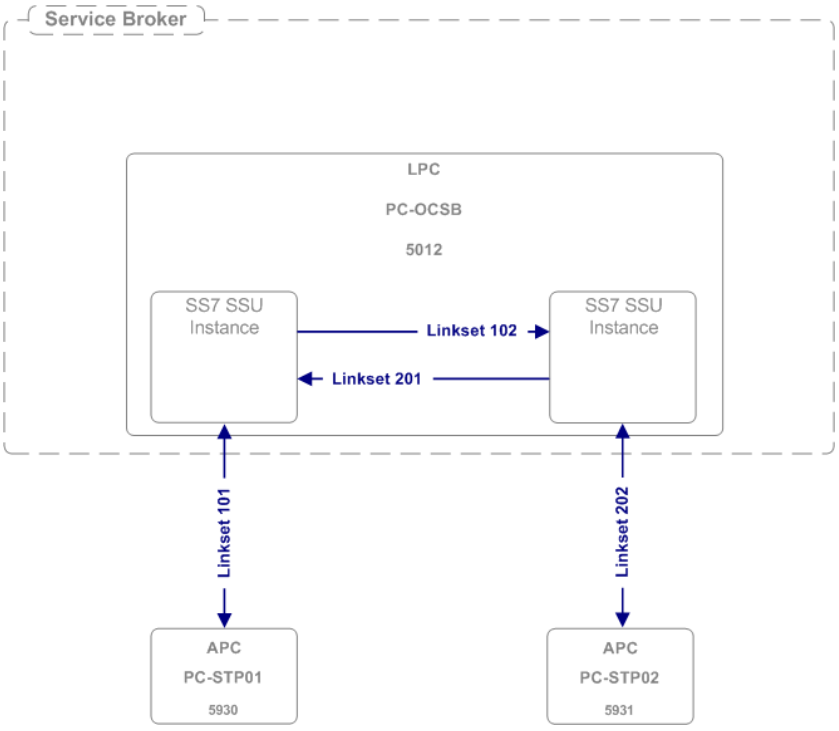
Subtab	Description
MtpLinkset	Enables you to configure MTP Linksets that connect SSU to adjacent signaling points. For more information, see "Configuring MTP Linksets".
MtpLink	Enables you to configure MTP links that connect SSU to adjacent signaling points. For more information, see "Configuring MTP Links".

Configuring MTP Linksets

The MtpLinkset subtab enables you to configure linksets for connecting an SSU to adjacent signaling points in an SS7 network.

Figure 3–3 shows an example of a linkset configuration.

Figure 3–3 Configuration Example: MTP Linkset



The MtpLinkset subtab contains a table in which each row represents one linkset. When defining a linkset, you need to specify the fields described in Table 3–9.

Table 3–9 MtpLinkset Fields

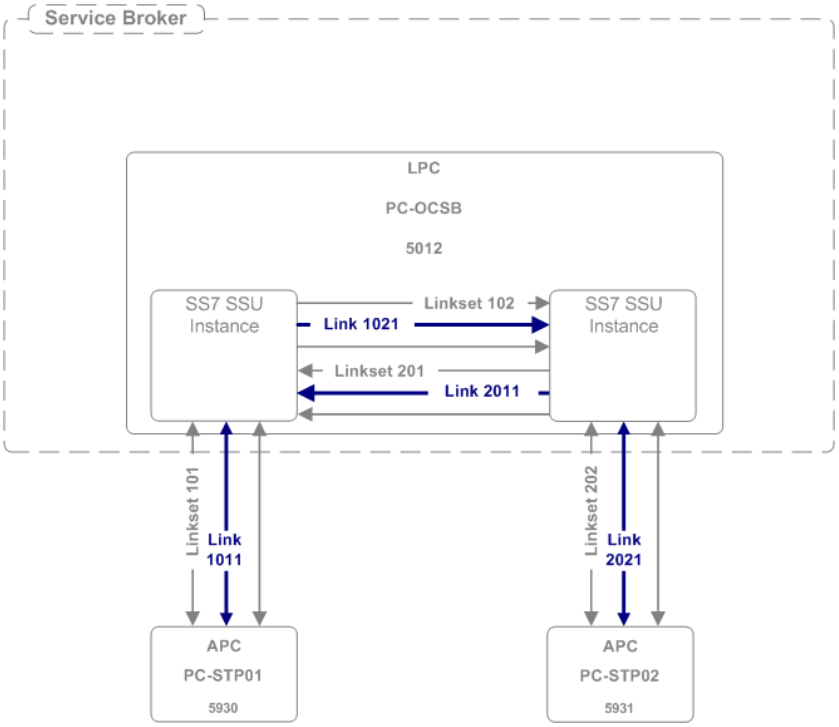
Name	Type	Description
Name	STRING	Specifies a unique linkset name.
Adjacent Point Code	INT	Specifies a point code on the far end of the linkset
Subservice	INT	Specifies a 4-bit value used in the Subservice field of all MTP3 messages that are passed over the linkset. This parameter can be set to any value from 0 to 16. Default value: 1

Configuring MTP Links

The MtpLink subtab enables you to configure links within linksets.

Figure 3–4 shows an example of a links configuration.

Figure 3–4 Configuration Example: MTP Links



The MtpLinks subtab contains a table in which each row represents one link. When defining a link, you need to specify the fields described in Table 3–10.

Table 3–10 MtpLink Fields

Name	Type	Description
Name	STRING	Specifies a unique link name.
PCM	STRING	Specifies a physical PCM cable with which a link is associated. You can select one of the PCMs that you have previously defined on the PCM subtab. (For more information on configuring PCMs, see "Configuring SS7 PCMs".)

Table 3–10 (Cont.) MtpLink Fields

Name	Type	Description
Time Slot	STRING	Specifies the PCMs time slot used for a signaling link. The value that you can select depends on the protocol you use: ■ E1: any value from 1 to 31 ■ T1: any value from 1 to 24 Default value: 1
Signaling Link Code	INT	Specifies a unique identifier (signaling link code) of the link in the linkset. The value must be unique within the link set. Signaling Link Code can be set to any value from 0 to 15. Default value: 0
Operation Mode	STRING	Specifies the rate on which the link operates. Possible values: ■ 56 kbits/s ■ 64 kbits/s Default value: 56 kbits/s
MTP2ECM	STRING	Specifies the MTP2 error correction mode. Possible values: ■ PCR: Preventive Cyclic Retransmission ■ BMEC: Basic Method of Error Correction Default value: BMEC

Network Routing

The Network Routing subtab enables you to configure how an SSU accesses SS7 network entities.

Table 3–11 describes subtabs in the Network Routing section.

Table 3–11 Network Routing Section Subtabs

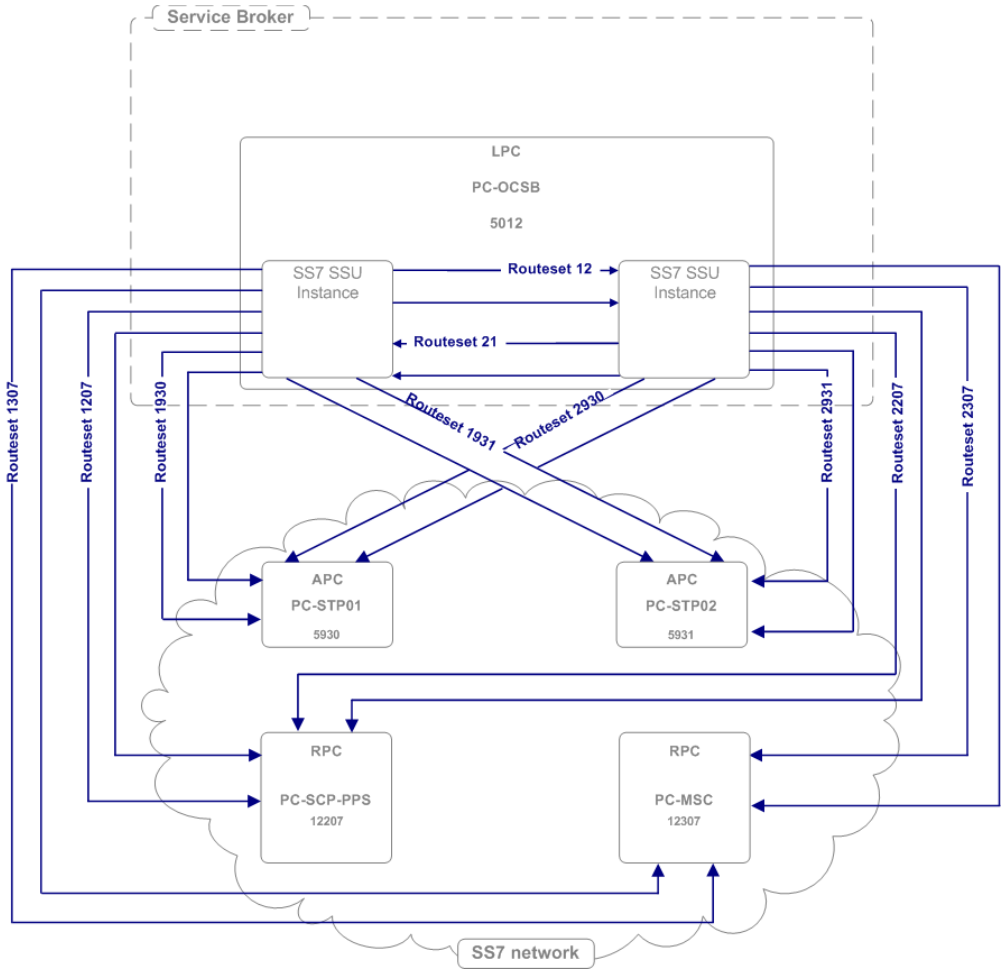
Subtab	Description
Routesets	Enables you to configure how an SSU instance accesses remote point codes in an SS7 network. For more information, see "Configuring Routesets".
Routes	Enables you to define the linkset that must be used to route a message to a remote point code. For more information, see "Configuring Routes".

Configuring Routesets

The Routesets subtab enables you to configure how an SSU instance accesses remote point codes in an SS7 network.

Figure 3–5 shows an example of a routesets configuration.

Figure 3–5 Configuration Example: MTP Routeset



The Routesets subtab contains a table in which each row represents a single routeset. When defining a routeset, you need to specify the fields described in Table 3–12.

Table 3–12 Routesets Fields

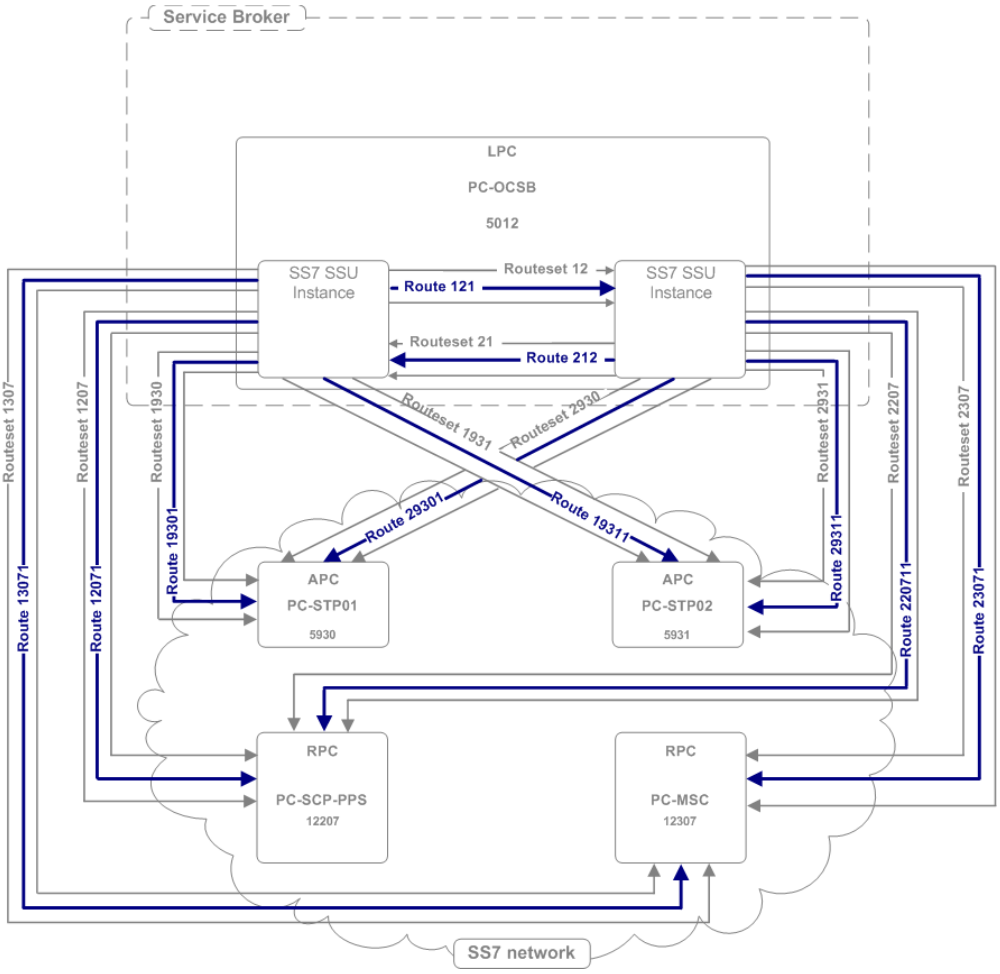
Name	Type	Description
Name	STRING	Specifies a unique routeset name.
Remote Point Code	INT	Specifies a point code or a remote SS7 entity
Default Route	STRING	Possible values: ■ Yes ■ No Default value: No
Description	STRING	Specifies a routeset description

Configuring Routes

The Routes subtab enables you to define routes within a routeset.

Figure 3–6 shows an example of a routes configuration.

Figure 3–6 Configuration Example: MTP Routes



The Routes subtab contains a table in which each row represents one route. When defining a route, you need to specify the fields described in Table 3–13.

Table 3–13 Routes Fields

Name	Type	Description
Name	STRING	Specifies a unique route name.
Primary Linkset	INT	Specifies a linkset over which messages are sent to a remote point code. You can select one of the linksets that you have previously defined on the MTP Linksets subtab. (For more information on configuring linksets, see "Configuring MTP Linksets".)
Secondary Linkset	INT	Specifies an alternative linkset over which messages can be sent to a remote point code. You can select one of the linksets that you have previously defined on the MTP Linksets subtab in the Network Mapping section. (For more information on configuring linksets, see "Configuring MTP Linksets".)

Table 3–13 (Cont.) Routes Fields

Name	Type	Description
Type	STRING	Specifies a route type. Possible values: ■ Standalone: An SSU sends messages to the RPC over the linkset specified in the Linkset parameter. The Standalone type cannot have an alternative linkset. ■ Preferred: An SSU sends messages to the RPC over the linkset specified in the Linkset parameter. If the sending messages over this fails, the SSU routes messages to the alternative linkset as defined in the SecondLinkset parameter. The Preferred type must have the Second Linkset defined. ■ Combined: An SSU sends messages to the RPC over both the linkset and alternative linkset as defined in the Linkset and Second Linkset parameters. The Combine type must have Second Linkset defined. Default value: Standalone

SCCP

The SCCP tab enables you to configure SCCP addresses for:

- Service Broker modules
- Remote entities in an SS7 network.

To access the SCCP tab:

- In the SSU SS7 TDM Configuration screen, click the SCCP tab.

  The SCCP configuration pane contains the subtabs described in Table 3–14.

Table 3–14 SCCP Section Subtabs

Subtab	Description
General	Enables you to specify parameters, which are common for all SCCP addresses. For more information, see "General".
Local SSNs	Enables you to assign subsystem numbers for Service Broker module instances. For more information, see "Local SSNs".
Local GTs	Enables you to configure Global Title addresses for Service Broker module instances. For more information, see "Local GTs".
Remote PC and SSN Addresses	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a point code and a subsystem number. For more information, see "Remote PC and SSN Addresses".
Remote Fixed GTs	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a fixed Global Title. For more information, see "Remote Fixed GTs".

Table 3–14 (Cont.) SCCP Section Subtabs

Subtab	Description
Remote Dynamic GTs	Enables you to configure addresses of remote entities in the SS7 network that can be reached using a dynamic Global Title. For more information, see "Remote Dynamic GTs".
Global Title Routing	Enables you to configure addresses of network entities that perform Global Title Translation. For more information, see "Global Title Routing".

General

The General subtab enables you to specify parameters, which are common for all SCCP addresses. Table 3–15 describes the parameters on the General subtab that you need to define.

Table 3–15 General Parameters

Name	Type	Description
Local Network Indicator	STRING	Specifies the network type of an SSU address, which is common for all SSU local SCCP addresses. Possible values: ■ International Network ■ International Network Extension ■ National Network ■ National Network Extension Default value: International Network The Local Network Indicator parameter of the M3UA stack is set to the same value as this parameter. However, because International Network Extension and National Network Extension are not supported in the M3UA stack, these two parameters are translated as follows in M3UA: ■ International Network Extension is translated to International Network ■ National Network Extension is translated to National Network
Remove Calling Party Point Code upon GT Routing	BOOL	Specifies whether or not the local SSU point code is to be added to the calling party address, when routing is done with a Global Title. Possible values: ■ True: the local point code is not added to the calling party address ■ False: the local point code is added to the calling party address

Table 3–15 (Cont.) General Parameters

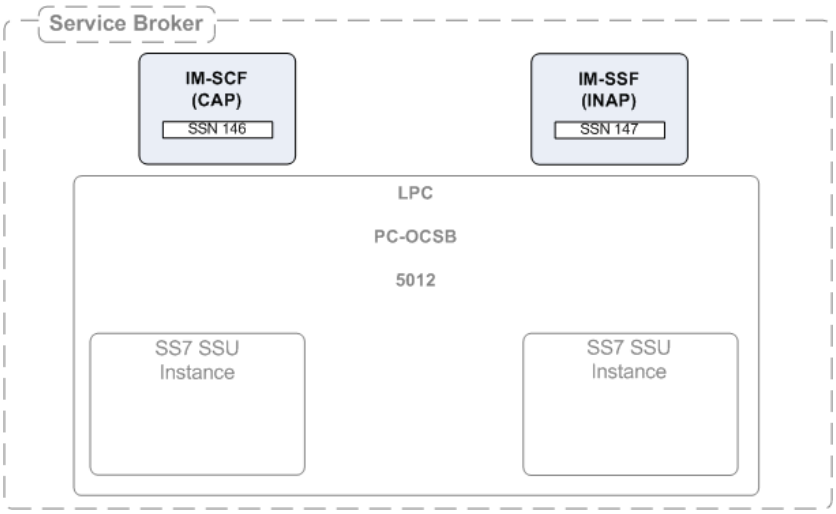
Name	Type	Description
Remove Called Party Point Code upon GT Routing	BOOL	Specifies whether or not the remote point code is to be removed from the called party address, when routing is done with a Global Title. Possible values: ■ True: the remote point code is not added to the called party address ■ False: the remote point code is added to the called party address

Local SSNs

The Local SSNs subtab enables you to assign Subsystem Numbers (SSNs) for Service Broker module instances. An SSU routes incoming messages to local subsystems based on these SSNs.

Figure 3–7 shows an example of a configuration of local SSNs.

Figure 3–7 Configuration Example: Local SSNs



The Local SSNs subtab contains a table in which each row represents a single Service Broker subsystem. When configuring an SSN, you need to specify the fields described in Table 3–16.

Table 3–16 Local SSNs Fields

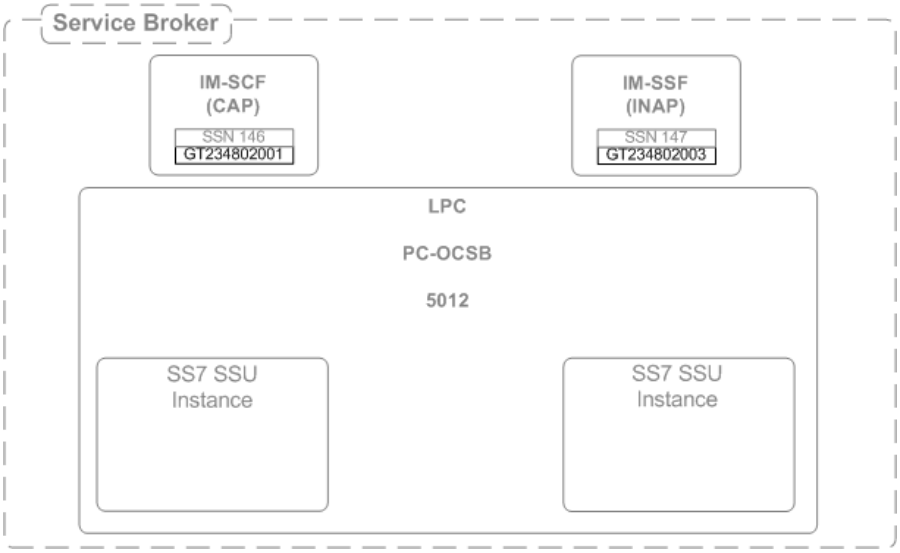
Name	Type	Description
Name	STRING	Specifies the subsystem name
SSN	INT	Specifies the subsystem number. Default value: 0.
Description	STRING	Specifies a subsystem description
Alias	STRING	Specifies an alias name given to a Service Broker subsystem. Applications that use Service Broker to connect to the SS7 network, use this alias to refer the specific subsystem.

Local GTs

The Local GTs subtab enables you to configure Global Title addresses for Service Broker module instances.

Figure 3–8 shows an example of a local GT configuration.

Figure 3–8 Configuration Example: Local GT



The Local GTs subtab contains a table in which each row represents a single address. When defining an address, you need to specify the fields described in Table 3–17.

Table 3–17 Local GTs Fields

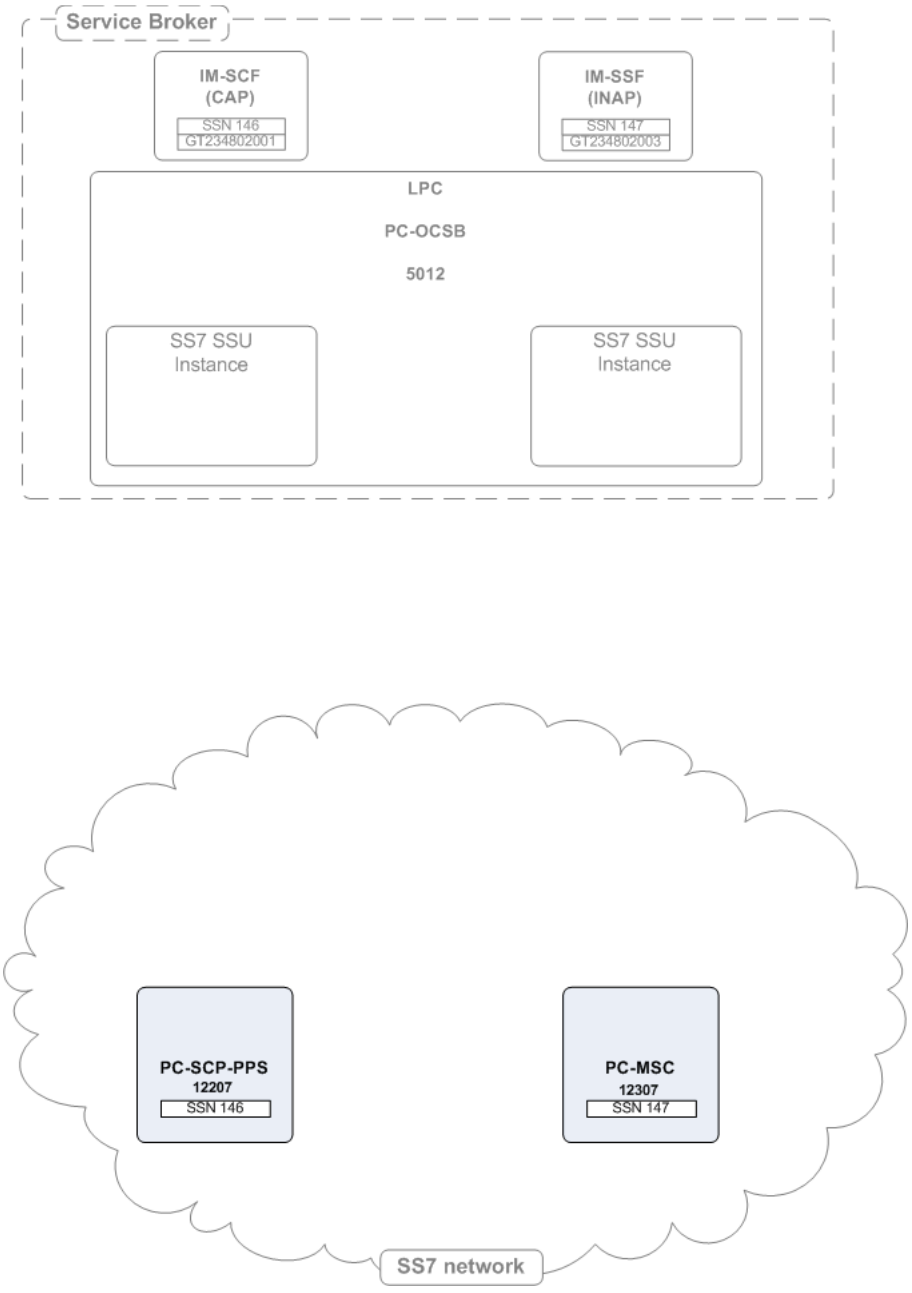
Name	Type	Description
Name	STRING	Specifies a unique name
Description	STRING	Specifies a description for the Service Broker GT address.
GT Address	STRING	Specifies the Global Title Address part of the SCCP address
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT.
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to a Service Broker subsystem. Applications that use Service Broker to connect to the SS7 network, use this alias to refer the specific GT address.

Remote PC and SSN Addresses

The Remote PC and SSN Addresses subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a point code and a subsystem number.

Figure 3–9 shows an example of a remote point code and SSN configuration.

Figure 3–9 Configuration Example: Remote PC and SSN



The Remote PC and SSN Addresses subtab contains a table in which each row represents a single SS7 network entity. When configuring a network entity, you need to specify the fields described in Table 3–18.

Table 3–18 Remote PC and SSN Fields

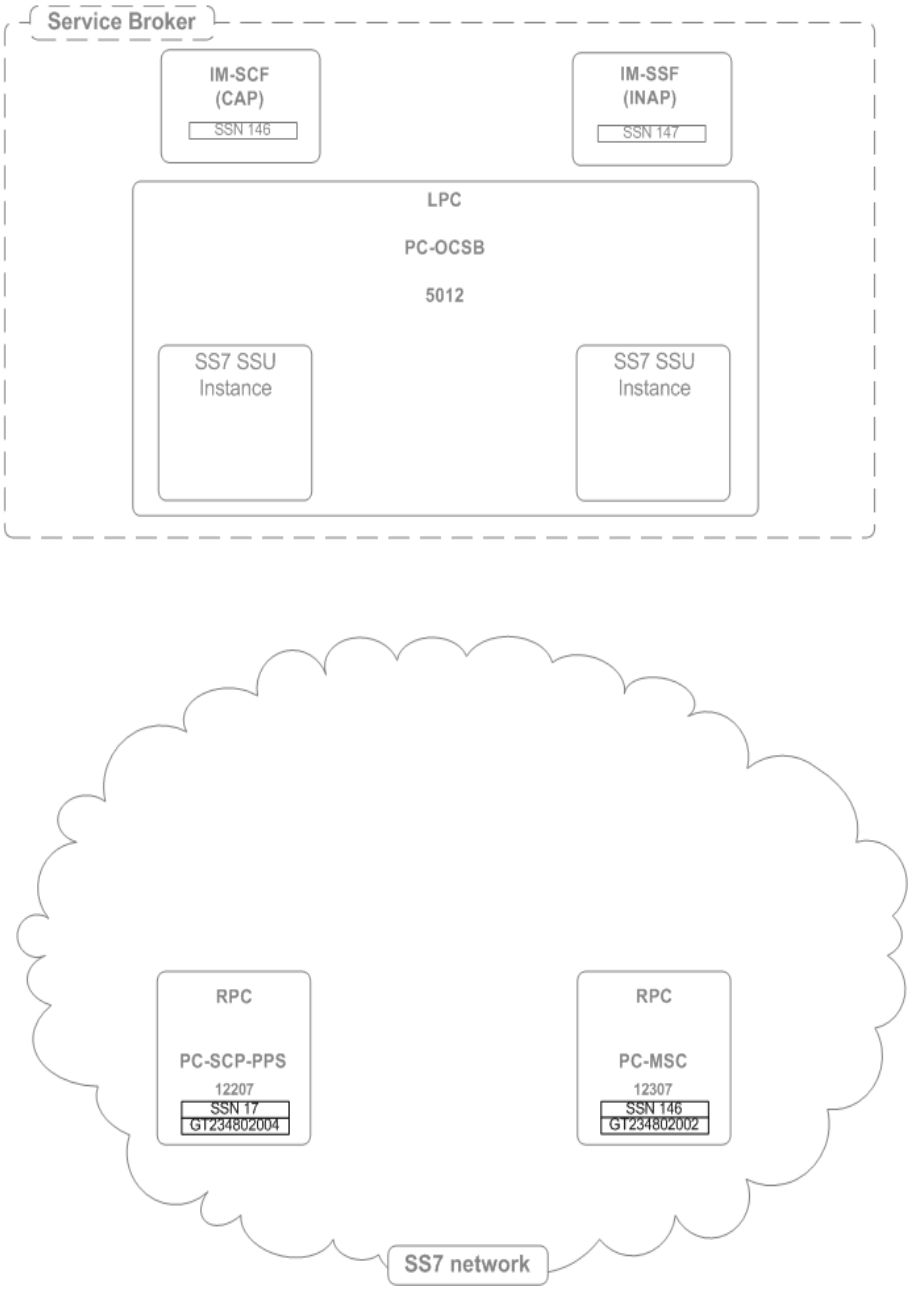
Name	Type	Description
Name	STRING	Specifies a unique name
Network Indicator	STRING	Specifies the network type. Possible values: ■ International Network ■ National Network Default value: International Network
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function.
Point Code	INT	Specifies the point code part of the SCCP address.
Description	STRING	Specifies a description for the remote SS7 network entity.
Alias	STRING	Specifies an alias name given to a remote network entity. Applications that use Service Broker to connect to the SS7 network, use this alias to refer the specific network entity.

Remote Fixed GTs

The Remote Fixed GTs subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a fixed Global Title.

Figure 3–10 shows an example of a remote fixed GTs configuration.

Figure 3–10 Configuration Example: Remote Fixed GTs



The Remote Fixed GTs subtab contains a table in which each row represents a single SS7 network entity. When configuring a network entity, you need to specify the fields described in Table 3–19.

Table 3–19 Remote Fixed GTs Fields

Name	Type	Description
Name	STRING	Specifies a unique name

Table 3–19 (Cont.) Remote Fixed GTs Fields

Name	Type	Description
Network Indicator	STRING	Specifies the network type. Possible options: ■ International Network ■ National Network Default option: International Network
Description	STRING	Specifies a description for the network entity and its address
GT Address	STRING	Specifies the Global Title Address part of the SCCP address
Point Code	INT	Optional: specifies the point code part of the SCCP address. When specified, the SSU routes messages to the specified point code, including a GT address.
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT.
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to a remote network entity. Applications that use Service Broker to connect to the SS7 network, use this alias to refer the specific network entity.

Remote Dynamic GTs

The Remote Dynamic GTs subtab enables you to configure addresses of remote entities in the SS7 network that can be reached using a dynamic Global Title.

The Remote Dynamic GTs subtab contains a table in which each row represents a single SCCP address. When configuring an SCCP address, you need to specify the fields described in Table 3–20.

Table 3–20 Remote Dynamic GTs Fields

Name	Type	Description
Name	STRING	Specifies a unique name
Network Indicator	STRING	Specifies the network type. The following options are available: ■ International Network ■ National Network Default value: International Network
Description	STRING	Specifies a description for the dynamic GT address
Point Code	INT	Optional: specifies the point code part of the SCCP address. When specified, the SSU routes messages to the specified point code, including a GT address.

Table 3–20 (Cont.) Remote Dynamic GTs Fields

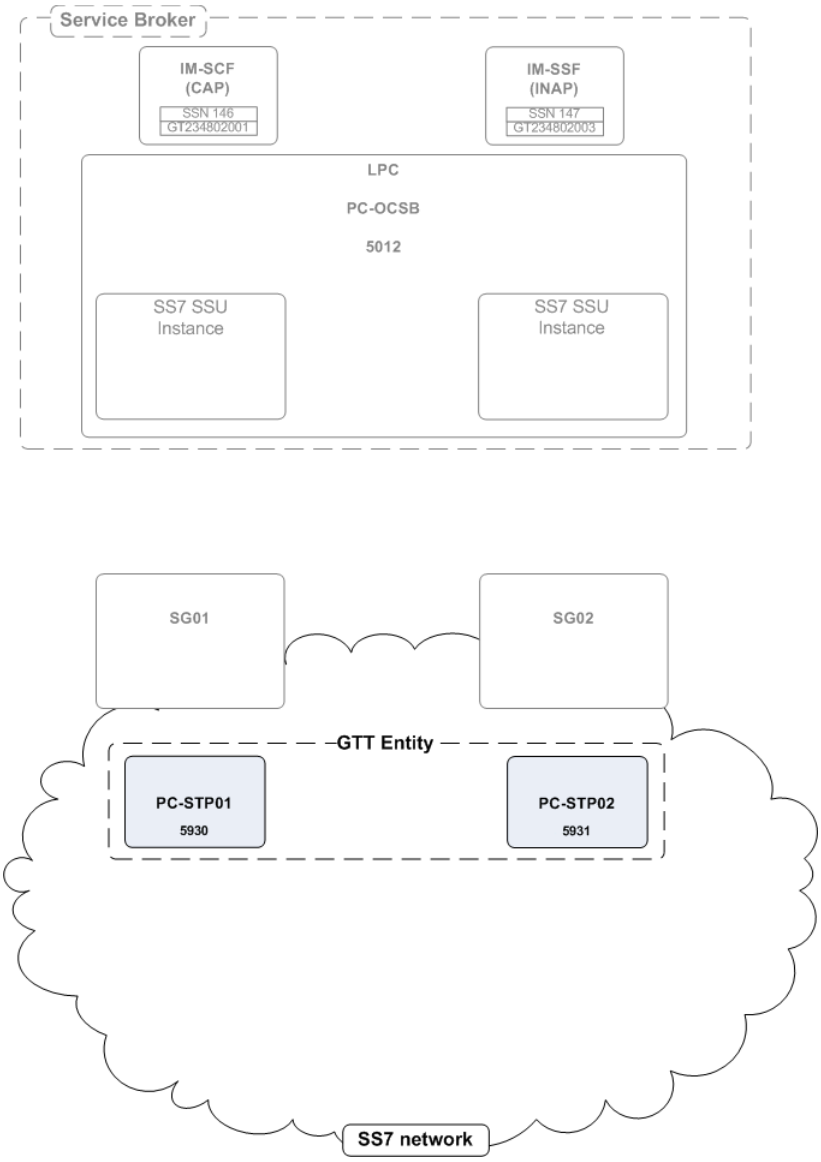
Name	Type	Description
SSN	INT	Specifies the SSN part of the SCCP address that identifies the user function
GT Indicator	INT	Specifies the Global Title Indicator part of the GT
GT Nature of Address	INT	Specifies the Nature of Address Indicator part of the GT
GT Numbering Plan	INT	Specifies the Numbering Plan part of the GT.
GT Translation Type	INT	Specifies the Translation Type part of the SCCP address
Alias	STRING	Specifies an alias name given to an SCCP address. Applications that use Service Broker to connect to the SS7 network, use this alias when they wish route messages using this address.

Global Title Routing

The Global Title Routing subtab enables you to configure addresses of network entities that perform Global Title Translation. Typically these point codes are Signal Transfer Points (STPs).

Figure 3–11 shows an example of a point code configuration.

Figure 3–11 Configuration Example: Global Title Routing



The Global Title Routing subtab contains a table in which each row represents a point code that performs GTT. When defining a point code that performs GTT, you need to specify the fields described in Table 3–21.

Table 3–21 Global Title Routing Parameters

Name	Type	Description
Primary GTT Point Code	INT	Specifies a primary remote point code that performs GTT.
Secondary GTT Point Code	INT	Specifies an alternative remote point code that performs GTT.

Table 3–21 (Cont.) Global Title Routing Parameters

Name	Type	Description
Operation Mode	STRING	Specifies the mode in which the primary and secondary remote point codes operate. The following options are available: ■ LOAD_SHARING: the SSU sends messages to both primary and secondary point codes in a load sharing mode. ■ PRIMARY_SECONDARY: the SSU sends messages to the primary point code. If the primary point code is not available, the SSU routes messages to the secondary point code. Default value: PRIMARY_SECONDARY

Routing

The Routing tab enables you to define an IM to which SS7 SSU routes an incoming session by specifying a set of parameters known as incoming routing rules. For each incoming routing rule, you need to configure the following parameters:

- IM to which SS7 SSU routes an incoming session
- Criteria that an incoming session must meet to be routed to this IM
- Priority in which SS7 SSU checks incoming routing rules to evaluate whether an incoming session fits the criteria defined in a rule. SS7 SSU applies the first found rule which criteria are met by an incoming session.

  For example, if you created multiple rules for the same IM, SS7 SSU begins with the rule that has the highest priority. If an incoming session fits the criteria defined in this rule, SS7 SSU applies the rule and do not check the rest of the rules. Otherwise, SS7 SSU checks whether an incoming session fits the criteria of a rule with a lower priority. SS7 SSU performs this check until SS7 SSU finds a rule whose criteria are met by an incoming session.

You can define incoming routing rules using the Routing tab. The process of defining an incoming routing rule consists of the following steps:

1. You create a rule and define its name, priority, and the IM for which you are creating the rule. You perform these actions using the Incoming Routing Rules subtab.
2. You define criteria for each rule that you created in step 1.

Accessing the Routing Tab

The Routing tab enables you to define rules for routing incoming sessions to IMs.

To access the Routing tab:

1. In the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Select **SSU SS7 TDM**.
4. In the configuration pane, click the **Routing** tab.

   The Routing pane contains the subtabs described in Table 3–22.

Table 3–22 Routing Subtabs

Subtab	Description
Incoming Routing Rules	Enables you to define a name, priority, and an IM for which you create a rule. For more information, see "Configuring Incoming Routing Rules Parameters".
Incoming Routing Criteria	Enables you to define criteria for each routing rule created on the Incoming Routing Rules subtab. For more information, see "Configuring Incoming Routing Criteria Parameters".

Configuring Incoming Routing Rules Parameters

The Incoming Routing Rules subtab enables you to define a name, priority, and an IM for which you create a rule. The Incoming Routing Rules subtab contains a table in which each row represents an individual rule.

When you define a rule, you need to specify the fields defined in Table 3–23.

Table 3–23 Incoming Routing Rule Fields

Name	Type	Description
Name	STRING	Specifies a unique rule name
Priority	INT	Specifies an order in which SS7 SSU checks routing rules to evaluate if an incoming session fits rule's criteria. SS7 SSU applies the first found rule which criteria are met by an incoming session. The lower the number, the higher the priority. For example, if you created two rules and set Priority of one rule to" 1" and set Priority of another rule to "2", SS7 SSU checks the rule with Priority set to "1" first. You can define an incoming routing rule that SS7 SSU applies if no other rule can be applied, by setting the Priority parameter of this rule to the largest number (that is lowest priority). There is no need to specify incoming routing criteria for such a rule.

Table 3–23 (Cont.) Incoming Routing Rule Fields

Name	Type	Description
Module Instance	STRING	Specifies the URI of an IM to which the SS7 SSU routes an incoming session. The URI has the following format: *IM-instance-name.IM-type@domain-id* ■ *IM-instance-name*: The IM instance name that you specified when you added this IM in the IM Management Configuration screen. ■ *IM-type*: The type of the IM instance ■ *domain-id*: The name of a Processing Domain or a Processing Domain Group where the relevant IM is deployed. See "Setting Up the Service Broker Domain Name" in the *Oracle Communications Service Broker System Administrator's Guide* for more information on setting up a domain name. To set a Processing Domain, you must specify the name you configured for the domain during its creation.For more information, see "Setting a Service Broker Domain Name" in *Oracle Communications Service Broker Processing Domain Configuration Guide*. To set a Processing Domain Group, you must specify the group name.For more information about Processing Domain Groups, see "Managing Processing Domain Groups" in *Oracle Communications Service Broker Processing Domain Configuration Guide*. Example: `imscfcap4_instance.IMSCFCAP4@processing-domain-1`

Configuring Incoming Routing Criteria Parameters

The Incoming Routing Criteria subtab enables you to define criteria for rules that you created on the Incoming Routing Rules subtab. The Incoming Routing Criteria contains a table in which each row represents a routing rule.

When you define criteria, you need to specify the fields defined in Table 3–24.

Table 3–24 Incoming Routing Criteria Fields

Name	Type	Description
Parent	STRING	Specifies the name of a rule for which you want to define criteria. This list contains names of the rules that you created on the Incoming Routing Rules subtab.
Name	STRING	Specifies a unique rule name
Session Key	STRING	Specifies a parameter inside an SCCP message based on which the SS7 SSU performs routing. The SS7 SSU will route incoming messages to a specified module instance, if the value of this parameter matches the Value field. Possible values: ■ DEST_ADDRESS_ALIAS ■ SOURCE_ADDRESS_ALIAS ■ APPLICATION_CONTEXT ■ SERVICE_KEY ■ OPCODE

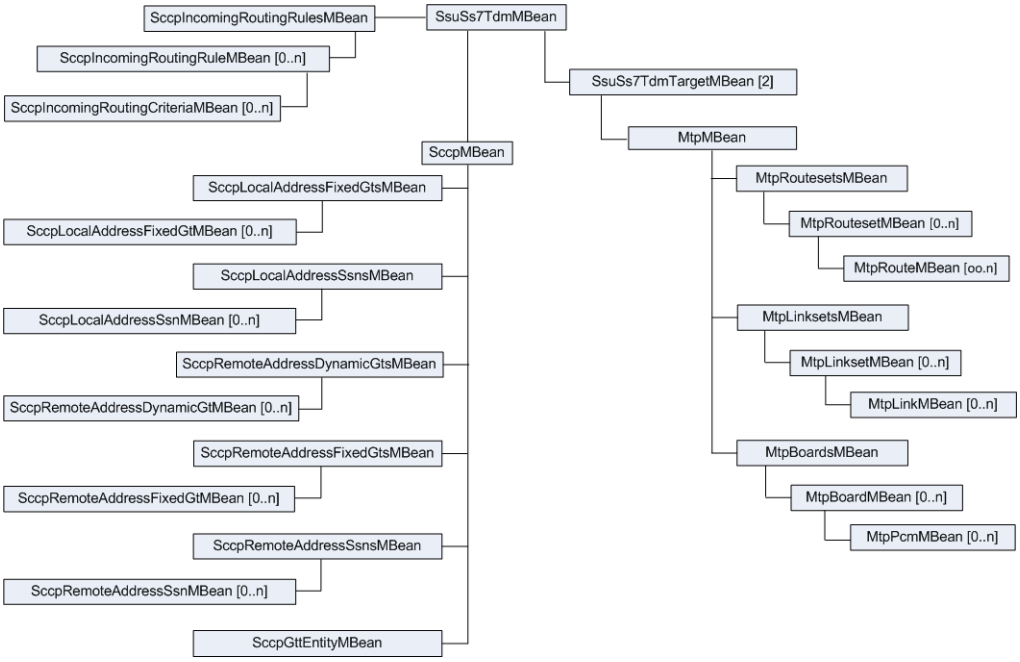
Table 3–24 (Cont.) Incoming Routing Criteria Fields

Name	Type	Description
Value	STRING	Specifies a value that the Session Key parameter of an SCCP message must match, in order for the rule specified in Parent to apply. You can define one of the following in the Value parameter: ▪ Single value ▪ Range of dash-separated values ▪ Comma-separated values

Configuring SSU SS7 TDM with Java MBeans

Service Broker provides a set of MBeans that expose attributes and operations for configuring SSU SS7 for TDM through JMX. Figure 3–12 shows the hierarchy of MBeans that you use for configuring SSU SS7 for TDM. Because a Service Broker deployment includes two SSU instances, you need to configure an SsuTargetMBean for each SSU instance separately.

Figure 3–12 SSU SS7 TDM Configuration MBean Hierarchy



The following sections provide reference information for the SS7 SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

SsuSs7TdmMBean

SsuSs7TdmMBean is a root MBean for the SSU SS7 TDM configuration.

Factory Method

Created automatically

Attributes

- BoardType
- Vendor
- Standard
- Mtp3Rpo
- RoutesetTest
- Ss7StackIp
- Ss7StackPort

For more information on these attributes, see Table 3–2.

Operations

ObjectName getSccp()

Gets a reference to the instance of SccpMBean

ObjectName[] getSsuSs7SigtranTargets()

Gets an array of references to instances of SsuSs7SigtranTargetMBean

ObjectName getSccpIncomingRoutingRules()

Gets a reference to the instance of SccpIncomingRoutingRulesMBean

ObjectName createSsuSs7TdmTarget()

Creates a new instance of SsuSs7TdmTargetMBean

void destroySsuSs7TdmTarget()

Destroys an existing instance of SsuSs7TdmTargetMBean

ObjectName lookupSsuSs7TdmTarget()

Returns the SsuSs7TdmTargetMBean of the SSU running on the specified target server

SsuSs7TdmTargetMBean

SsuSs7TdmTargetMBean enables you to access the configuration MBeans of a specific SSU.

Factory Method

SsuSs7Tdm.createSsuSs7TdmTarget()

Attributes

None

Operations

ObjectName getMtp()

Returns a reference to the instance of MtpMBean

MtpMBean

MtpMBean enables you to configure MBeans of a specific SS7 SSU instance.

Factory Method

Created automatically

Attributes

LocalPointCode

For more information, see Table 3–4.

Operations

ObjectName getMtpBoards()

Returns a reference to the instance of MtpBoardsMBean

ObjectName getMtpLinksets()

Returns a reference to the instance of MtpLinksetsMBean

ObjectName getMtpRoutesets()

Returns a reference to the instance of MtpRoutesetsMBean

MtpBoardsMBean

MtpBoardsMBean is a root MBean for individual instances of MtpBoardMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getMtpBoards()

Gets an array of references to instances of MtpBoardsMBean

ObjectName createMtpBoard()

Creates a new instance of MtpBoardMBean

void destroyMtpBoard()

Destroys an existing instance of MtpBoardMBean

ObjectName lookupMtpBoard()

Returns a reference to the specified instance of MtpBoardMBean

MtpBoardMBean

MtpBoardMBean enables you to configure SS7 boards plugged into machine chassis for each SSU instance.

Factory Method

MtpBoards.createMtpBoard()

Attributes

- Name
- Slot
- Clock
- MtpPcm

For more information on these attributes, see Table 3–6.

Operations

ObjectName createMtpPcm()

Creates a new instance of MtpPcmMBean

ObjectName destroyMtpPcm()

Destroys an existing instance of MtpPcmMBean

ObjectName lookupMtpPcm()

Returns a reference to the specified instance of MtpPcmMBean

MtpPcmMBean

MtpPcmMBean enables you to configure a PCM that physically connect the SSU instances to the SS7 network.

Factory Method

MtpBoard.createMtpPcm()

Attributes

- Name
- Type
- Crc
- Code
- Frame
- Port

For more information on these attributes, see Table 3–7.

Operations

None

MtpLinksetsMBean

MtpLinksetsMBean is a root MBean for individual instances of MtpLinksetMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getMtpLinksets()

Gets an array of references to instances of MtpLinksetsMBean

ObjectName createMtpLinkset()

Creates a new instance of MtpLinksetMBean

void destroyMtpLinkset()

Destroys an existing instance of MtpLinksetMBean

ObjectName lookupMtpLinkset()

Returns a reference to the specified instance of MtpLinksetMBean

MtpLinksetMBean

MtpLinksetMBean enables you to configure linksets for connecting SS7 SSUs to adjacent signaling points in an SS7 network.

Factory Method

MtpLinksets.createMtpLinkset()

Attributes

- Name
- AdjacentPoint Code
- Subservice
- MtpLinks

For more information on these attributes, see Table 3–9.

Operations

ObjectName createMtpLink()

Adds an MtpLink to the Linkeset

void destroyMtpLink()

Removes an MtpLink from the Linkset

ObjectName lookupMtpLink()

Lookups for an MtpLink

MtpLinkMBean

MtpLinkMBean enables you to configure links within a linkset

Factory Method

MtpLinkset.createMtpLink()

Attributes

- Name
- Pcm
- TimeSlot
- SignalingLinkCode
- OperationMode
- Mtp2Ecm

For more information on these attributes, see Table 3–10.

Operations

None

MtpRoutesetsMBean

MtpRoutesetsMBean is a root MBean for individual instances of MtpRoutesetMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getMtpRoutesets()

Gets an array of references to instances of MtpRoutesetsMBean

ObjectName createMtpRouteset()

Creates a new instance of MtpRoutesetMBean

void destroyMtpRouteset()

Destroys an existing instance of MtpRoutesetMBean

ObjectName lookupMtpRouteset()

Returns a reference to the specified instance of MtpRoutesetMBean

MtpRoutesetMBean

MtpRoutesetMBean enables you to configure how an SS7 SSU instance can access remote point codes in an SS7 network.

Factory Method

MtpRoutesets.createMtpRouteset()

Attributes

- Name
- RemotePointCode
- DefaultRoute
- MtpRoutes
- Description

For more information on these attributes, see Table 3–12.

Operations

ObjectName createMtpRoute()

Creates an instance of MtpRoute

ObjectName destroyMtpRoute()

Destroys an instance of MtpRoute

ObjectName lookupMtpRoute()

Returns a reference to the specified instance of MtpRoute

MtpRouteMBean

MtpRouteMBean enables you to define the linkset that must be used to route the message to an accessible RPC.

Factory Method

MtpRouteset.createMtpRoute()

Attributes

- Name
- PrimaryLinkset
- SecondaryLinkset
- Type

For more information on these attributes, see Table 3–13.

Operations

None

SccpIncomingRoutingRulesMBean

SccpIncomingRoutingRulesMBean enables you to define how the SS7 SSU routes incoming SS7 messages to internal Service Broker IMs.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpIncomingRoutingRules()

Gets an array of references to instances of SccpIncomingRoutingRuleMBean

ObjectName createSccpIncomingRoutingRule()

Creates a new instance of SccpIncomingRoutingRuleMBean

void destroySccpIncomingRoutingRule()

Destroys an existing instance of SccpIncomingRoutingRuleMBean

ObjectName lookupSccpIncomingRoutingRule()

Returns a reference to the SccpIncomingRoutingRuleMBean

SccpIncomingRoutingRuleMBean

SccpIncomingRoutingRuleMBean represents an individual incoming routing rule.

Factory Method

SccpIncomingRoutingRules.createSccpIncomingRoutingRule()

Attributes

- Name
- Priority
- ModuleInstance

For more information on these attributes, see Table 3–23.

Operations

ObjectName[] getSccpIncomingRoutingCriteria()

Gets an array of references to instances of SccpIncomingRoutingCriteriaMBean

ObjectName createSccpIncomingRoutingCriteria()

Creates a new instance of SccpIncomingRoutingCriteriaMBean

void destroySccpIncomingRoutingCriteria()

Destroys an existing instance of SccpIncomingRoutingCriteriaMBean

ObjectName lookupSccpIncomingRoutingCriteria()

Returns a reference to the SccpIncomingRoutingCriteriaMBean

SccpIncomingRoutingCriteriaMBean

SccpIncomingRoutingCriteriaMBean represents criteria for an individual incoming routing rule.

Factory Method

SccpIncomingRoutingRule.createSccpIncomingRoutingCriteria()

Attributes

- Parent
- Name
- SessionKey
- Value

For more information on these attributes, see Table 3–24.

Operations

None

SccpMBean

SccpMBean enables you to configure SCCP addresses: subsystems and global titling.

Factory Method

Created automatically

Attributes

- LocalNetworkIndicator
- RemoveCallingPartyPointCodeUponGtRouting
- RemoveCalledPartyPointCodeUponGtRouting

For more information, see Table 3–15.

Operations

ObjectName getSccpGttEntity()

Gets a reference to the instance of SccpGttEntityMBean

ObjectName[] getSccpLocalAddressFixedGts()

Gets an array of references to instances of SccpLocalAddressFixedGtMBean

ObjectName[] getSccpLocalAddressSsns()

Gets an array of references to instances of SccpLocalAddressSsnMBean

ObjectName[] getSccpRemoteAddressDynamicGts()

Gets an array of references to instances of SccpRemoteAddressDynamicGtMBean

ObjectName[] getSccpRemoteAddressFixedGts()

Gets an array of references to instances of SccpRemoteAddressFixedGtMBean

ObjectName[] getSccpRemoteAddressSsns()

Gets an array of references to instances of SccpRemoteAddressSsnMBean

SccpLocalAddressFixedGtsMBean

SccpLocalAddressFixedGtsMBean is a root MBean for individual instances of SccpLocalAddressFixedGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpLocalAddressFixedGts()

Gets an array of references to instances of SccpLocalAddressFixedGtMBean

ObjectName createSccpLocalAddressFixedGt()

Creates a new instance of SccpLocalAddressFixedGtMBean

void destroySccpLocalAddressFixedGt()

Destroys an existing instance of SccpLocalAddressFixedGtMBean

ObjectName lookupSccpLocalAddressFixedGt()

Returns a reference to the specified instance of SccpLocalAddressFixedGtMBean

SccpLocalAddressFixedGtMBean

SccpLocalAddressFixedGtMBean enables you to configure a Global Title address for a Service Broker module instance.

Factory Method

SccpLocalAddressFixedGts.createSccpLocalAddressFixedGt()

Attributes

- Name
- Description
- GtAddress
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 3–17.

Operations

None

SccpLocalAddressSsnsMBean

SccpLocalAddressSsnsMBean is a root MBean for individual instances of SccpLocalAddressSsnMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpLocalAddressSsns()

Gets an array of references to instances of SccpLocalAddressSsnMBean

ObjectName createSccpLocalAddressSsn()

Creates a new instance of SccpLocalAddressSsn

void destroySccpLocalAddressSsn()

Destroys an existing instance of SccpLocalAddressSsn

ObjectName lookupSccpLocalAddressSsn()

Returns a reference to the specified instance of SccpLocalAddressSsn

SccpLocalAddressSsnMBean

SccpLocalAddressSsnMBean enables you to assign a subsystem number to a Service Broker module instance.

Factory Method

SccpLocalAddressSsns.createSccpLocalAddressSsn()

Attributes

- Name
- Ssn
- Description
- Alias

For more information on these attributes, see Table 2–12.

Operations

None

SccpRemoteAddressDynamicGtsMBean

SccpRemoteAddressDynamicGtsMBean is a root MBean for individual instances of SccpRemoteAddressDynamicGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressDynamicGts()

Gets an array of references to instances of SccpRemoteAddressDynamicGtMBean

ObjectName createSccpRemoteAddressDynamicGt()

Creates a new instance of SccpRemoteAddressDynamicGtMBean

void destroySccpRemoteAddressDynamicGt()

Destroys an existing instance of SccpRemoteAddressDynamicGtMBean

ObjectName lookupSccpRemoteAddressDynamicGt()

Returns a reference to the specified instance of SccpRemoteAddressDynamicGtMBean

SccpRemoteAddressDynamicGtMBean

SccpRemoteAddressDynamicGTMBean enables you to configure an address of a remote entity in the SS7 network that can be reached using a dynamic Global Title.

Factory Method

SccpRemoteAddressDynamicGts.createSccpRemoteAddressDynamicGt()

Attributes

- Name
- NetworkIndicator
- Description
- PointCode
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 2–16.

Operations

None

SccpRemoteAddressFixedGtsMBean

SccpRemoteAddressFixedGtsMBean is a root MBean for individual instances of SccpRemoteAddressFixedGtMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressFixedGts()

Gets an array of references to instances of SccpRemoteAddressFixedGtMBean

ObjectName createSccpRemoteAddressFixedGt()

Creates a new instance of SccpRemoteAddressFixedGtMBean

void destroySccpRemoteAddressFixedGt()

Destroys an existing instance of SccpRemoteAddressFixedGtMBean

ObjectName lookupSccpRemoteAddressFixedGt()

Returns a reference to the specified instance of SccpRemoteAddressFixedGtMBean

SccpRemoteAddressFixedGtMBean

SccpRemoteAddressFixedGTMBean enables you to configure an address of a remote entity in the SS7 network that can be reached using a fixed Global Title.

Factory Method

SccpRemoteAddressFixedGts.createSccpRemoteAddressFixedGt()

Attributes

- Name
- NetworkIndicator
- Description
- GtAddress
- PointCode
- Ssn
- GtIndicator
- GtNatureOfAddress
- GtNumberingPlan
- GtTranslationType
- Alias

For more information on these attributes, see Table 2–15.

Operations

None

SccpRemoteAddressSsnsMBean

SccpRemoteAddressSsnsMBean is a root MBean for individual instances of SccpRemoteAddressSsnMBean.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSccpRemoteAddressSsns()

Gets an array of references to instances of SccpRemoteAddressSsnMBean

ObjectName createSccpRemoteAddressSsn()

Creates a new instance of SccpRemoteAddressSsnMBean

void destroySccpRemoteAddressSsn()

Destroys an existing instance of SccpRemoteAddressSsnMBean

ObjectName lookupSccpRemoteAddressSsn()

Returns a reference to the specified instance of SccpRemoteAddressSsnMBean

SccpRemoteAddressSsnMBean

SccpRemoteAddressSsnMBean enables you to configure an address of a remote entity in the SS7 network, that can be reached using a point code and a subsystem number.

Factory Method

SccpRemoteAddressSsns.createSccpRemoteAddressSsn()

Attributes

- Name
- NetworkIndicator
- Ssn
- PointCode
- Description
- Alias

For more information on these attributes, see Table 3–12.

Operations

None

SccpGttEntityMBean

SccpGttEntityMBean enables you to configure an address of a network entity that performs Global Title Translation.

Factory Method

Created automatically

Attributes

- PrimaryPointCode
- SecondaryPointCode
- OperationMode

For more information on these attributes, see Table 3–21.

Operations

None

4

Configuring SIP Signaling Server Units

The following sections describe how to configure Service Broker SIP SSUs using the Service Broker Administration Console and Java MBeans.

Configuring SIP SSU with the Administration Console

Configuring SIP SSUs involves performing the tasks described in Table 4–1.

Table 4–1 SIP Signaling Server Configuration Tasks

Task	Description
SIP Server	Specifies a Globally-Routable User Agent URI (GRUU). See "Configuring the SIP Server" for more information.
Incoming Routing Rules	Specifies how the SIP SSU routes incoming SIP messages to internal Service Broker IMs and to applications external to OCSB. See "Configuring Incoming Routing Rules" for more information.
SIP Network Entities	Specifies a list of SIP network entities with which the SIP SSU communicate. See "Configuring SIP Network Entities" for more information.
Connection Pool	Specifies the connection that can be used to minimize communication overhead with SIP entities, such as Serving Call Session Control Function (S-CSCF). See "Configuring the SIP Connection Pool" for more information.
Network Access Points	Specifies the SIP network channels that the SIP SSU uses to listen to incoming connections. See "Configuring SIP Network Channels" for more information.
Header Insertion	Specifies how the SIP SSU inserts headers into SIP messages. See "Configuring Header Insertion Fields" for more information.

Configuring the SIP Server

You use the **SIP Server** tab to set up a Globally-Routable User Agent URI (GRUU).

To set up a GRUU:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.

3. In the SSU SIP configuration pane, click the **SSU SIP** tab and then the **SIP Server** subtab.
4. In the **Globally Routable User Agent URI** field, specify a SIP URI that the SIP SSU automatically inserts into Contact and Routeset headers when communicating with network elements.

   For example: `sip:sb@209.95.109.191:5060`.
5. Click **Apply**.

Configuring Incoming Routing Rules

You use the **Incoming Routing Rules** tab to define how the SIP SSU routes incoming SIP messages to internal Service Broker IMs. The SIP SSU can also be used to route incoming SIP messages to applications that are external to OCSB, according to an application ID specified by the application developer.

To configure incoming routing rules for SIP messages:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.
3. In the SSU SIP configuration pane, click the **SSU SIP** tab and then the **Incoming Routing Rules** subtab.

   The Incoming Routing Rules pane contains a table in which each row represents one routing rule.
4. Click the **New** button, located at the bottom of the Incoming Routing Rules pane.

   The New dialog box appears.
5. Fill in the fields in the dialog box described in Table 4–2.

Table 4–2 SIP Incoming Routing Rules Parameters

Field	Description
Name	Specifies a unique routing rule name
IP Address	Specifies an IP address by which the Service Broker IM is known to external entities. The SIP SSU routes incoming messages destined for this IP address to the IM, based on the name you specify in the **Alias** field. Setting this field to **any** applies the routing rule to any incoming SIP message, regardless of its destination IP address. This value is generally used for a default routing rule that routes the incoming SIP message to the default IM, in the absence of any other routing rule. Important: When typing **any** into the IP Address field, you must use only lower case, as follows: any. Do not type Any or ANY.

Table 4–2 (Cont.) SIP Incoming Routing Rules Parameters

Field	Description
Alias	Specifies the SIP URI of the IM to which the SIP SSU routes an incoming session. The alias has the following format: `SSU:`*IM-instance-name.IM-type@domain-id* ■ *IM-instance-name*: IM instance name you specified when you added this IM in the IM configuration pane. ■ *IM-type*: Type of IM instance. ■ *domain-id*: Name of the Processing Domain or Processing Domain Group where the relevant IM is deployed. This parameter is required only when your Service Broker deployment includes two or more Processing Domains. Use the name given to the domain when it was created. This name is specified by the axia.domain.id property. To set a Processing Domain Group, you must specify the group name. See "Managing Processing Domain Groups" in the *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information. Example: `ssu:r-imocf_instance.RIMOCF@processing-domain.1` You can also specify the SIP URI of an application external to OCSB to which the SIP SSU routes an incoming message. The alias has the following format: `SSU:`*parameter*`@processing-domain.1/`*application* ■ *parameter*: Developer-specfied parameters that are configured to forward the incoming message to the specified application ■ *application*: The destination application registered to the specified ID to which the incoming event is dispatched.

6. Click **OK**.

   The values you enter are displayed in the table.

Configuring SIP Network Entities

You use the **SIP Network Entities** tab to define a list of SIP network entities with which the SIP SSU communicates.

The **SIP Network Entities** tab contains a table in which each row represents one SIP network entity.

To provide a stable connection with a SIP network entity, the SIP SSU regularly checks the connection with the entity using a heartbeat mechanism. The heartbeat mechanism enables the SIP SSU to check whether a SIP network entity is active or inactive by periodically sending requests to the SIP network entity and checking the returned response.

After the SIP SSU triggers the heartbeat method, one of the following occurs:

- The SIP network entity responds within the time defined in **Response Timeout**. This means that the SIP network entity is active. The SIP SSU continues triggering the heartbeat method periodically, as specified by **Active Interval**.
- The SIP network entity does not respond within the time defined in **Response Timeout**. This means the SIP network entity is inactive. The SIP SSU continues triggering the heartbeat method as specified by **Inactive Interval**.

To configure SIP Network Entities:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.
3. In the SSU SIP configuration pane, click the **SSU SIP** tab and then the **SIP Network Entities** subtab.
4. Click the **New** button at the bottom of the **SIP Network Entities** pane.

   The New dialog box appears.
5. Fill in the fields described in Table 4–3.

Table 4–3 SIP Network Entities Parameters

Field	Description
Name	Specifies a unique network entity name.
Alias	Specifies a name that Service Broker uses to refer to the SIP network entity. The alias has a format of a SIP URI. For example: `sip:simple_b2b@example.com`
Heartbeat	Specifies whether or not to use a heartbeat mechanism over the connection with the SIP network entity. Possible values: ▪ ON ▪ OFF Default value: ON
SipUri	Specifies the SIP URI of the SIP network entity. For example: `sip:simple_b2b@192.168.0.219:6060`
Heartbeat Method	Specifies the SIP method that the SIP SSU uses to test the connection with the SIP network entity. Default value: OPTIONS
Response Timeout	Specifies the time interval in seconds during which the SIP SSU waits for a response from the SIP network entity. This field is used by the heartbeat mechanism.
Active Interval	Specifies the time interval in seconds for sending heartbeat requests from the SIP SSU to the SIP network entity. This field is used if the previous heartbeat test showed that the SIP network entity is active.
Inactive Interval	Specifies the time interval in seconds for sending heartbeat requests from the SIP SSU to the SIP network entity. This field is used if the previous heartbeat test showed that the SIP network entity is inactive.

6. Click **OK**.

   The values you enter are displayed in the table.

Configuring the SIP Connection Pool

Service Broker includes a connection pooling mechanism that can be used to minimize communication overhead with SIP entities, such as Serving Call Session Control

Function (S-CSCF). You can configure multiple, fixed pools of connections to different addresses.

Service Broker opens new connections from the connection pool on demand as the server makes requests to a configured address. The server then multiplexes new SIP requests to the address using the already-opened connections, rather than repeatedly terminating and recreating new connections. Opened connections are re-used in a round-robin fashion. Opened connections remain open until they are explicitly closed by the remote address.

Note that connection pools are not used for incoming requests from a configured address.

To configure a connection re-use pool for the network access point:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.
3. In the SSU SIP configuration pane, click the **SIP** tab and then the **Connection Pool** subtab.

   The Connection Pool pane contains a table in which each row represents a pool of connections to one destination host.
4. Click the **New** button (+) at the bottom of the Connection Pool configuration pane.

   The New Data dialog box appears.
5. Fill in the fields described in Table 4–4.

Table 4–4 Connection Pool Parameters

Field	Description
Name	Specifies a string value that identifies the name of this pool. All configured names must be unique to the domain.
Destination Host	Specifies the IP address or host name of the destination. Service Broker opens or re-uses connections in this pool only when making requests to the configured address.
Destination Port	Specifies the destination port number.
Maximum Connections	Specifies the maximum number of opened connections to maintain in the pool.

6. Click **OK**.

   The values you enter are displayed in the table.

Configuring SIP Network Channels

You use the **Network Access Points** tab to define SIP network channels that the SIP SSU uses to listen to incoming connections. **Network Access** Points contains the following subtabs that allow you to specify parameters representing a single SIP network channel:

- **General**
- **Listen address**
- **External listen address**

To configure a SIP Network Channel:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.
3. In the SSU SIP configuration pane, click the **SIP** tab and then click the **Network Access Point** subtab.
4. Do one of the following:
   - In the Network Access point pane, select the SSU instance whose network channel you want to configure in the Network Access Point tree
   - Click **Create a new node** (+) at the bottom of the pane to add a new SSU instance

   The listening address for each instance must be set to the corresponding blade's SIP IP address, as specified in the Network configuration pane. For more information, see "Connecting to the Network" in *Oracle Communications Service Broker Netra 6000 High Availability Manager Administrator's Guide*.
5. Click the **General** tab and enter values for the parameters described in Table 4–5.

Table 4–5 SIP Network Access Point General Parameters

Field	Description
Name	A unique SIP network channel name.
Protocol	The protocol used for connections through the network channel. Set this field to **SIP**.
Target	The name of the Signaling Server (JVM) providing the SIP network channel. Leaving this field blank indicates that the configuration applies to all Signaling Servers.
Maximum Connected Clients	Specifies the maximum number of SIP clients that can connect.
Complete Message Timeout	Specifies the amount of time in seconds the network access point waits for a complete message to be received. A value of **0** disables the network access point complete message timeout. Valid values range from **0** to **480** seconds.
Idle Connection Timeout	Specifies the amount of time in seconds that a connection is allowed to be idle before it is closed by the network access point. The minimum value is **0** seconds.

6. Click the **Listen Address** subtab, and enter values for the parameters described in Table 4–6.

Table 4–6 SIP Network Access Point Listen Parameters

Field	Description
Address Type	Specifies the address type of the SIP network channel internal identity. Set this field to: **IP4**
Host	Specifies the IP address or DNS name of the SIP network channel internal identity. Setting the value to **0.0.0.0** resolves to the IP of the local machine.
Network Type	Specifies the network type of the SIP network channel internal identity. Set this field to: **internet**
Port	Specifies the port of the SIP network channel internal identity.

7. Click the **External Listen Address** subtab and enter values for the parameters described in Table 4–7.

Table 4–7 SIP Network Access Point External Listen Address Parameters

Field	Description
Address Type	Specifies the address type of the SIP network external identity. Set this field to **IP4**
Host	Specifies the IP address or DNS name of the SIP network channel external identity. To set the host to the one specified in Table 4–6, set this parameter to **0.0.0.0**
Network Type	Specifies the network type of the SIP network channel external identity. Set this field to **internet**
Port	Specifies the port of the SIP network channel internal identity

Configuring Header Insertion Fields

You use the Network Access pane to set the SIP header insertion fields.

To configure the SIP header insertion fields:

1. In the navigation tree in the domain navigation pane, expand the **Signaling Tier** node.
2. Select **SSU SIP**.
3. In the SSU SIP configuration pane, click the **SIP** tab.
4. Fill in the fields described in Table 4–8.

Table 4–8 SIP Header Insertion Parameters

Field	Description
Default Form For Header Insertion field	Specifies how the SIP SSU applies rules for compacting SIP message headers. Possible values are: ■ Compact ■ Force compact ■ Long ■ Force long
Server Header Insertion	Specifies the conditions for inserting a server header in SIP messages. Possible values are: ■ None (default) ■ Request ■ Response ■ All
Server Header Value	Specifies the value of the server header inserted into SIP messages.

Configuring SIP SSU with Java MBeans

Figure 4–1 and Figure 4–2 show a set of MBeans that expose attributes and operations for configuring SIP SSU through JMX.

Figure 4–1 SIP SSU Configuration MBean Hierarchy

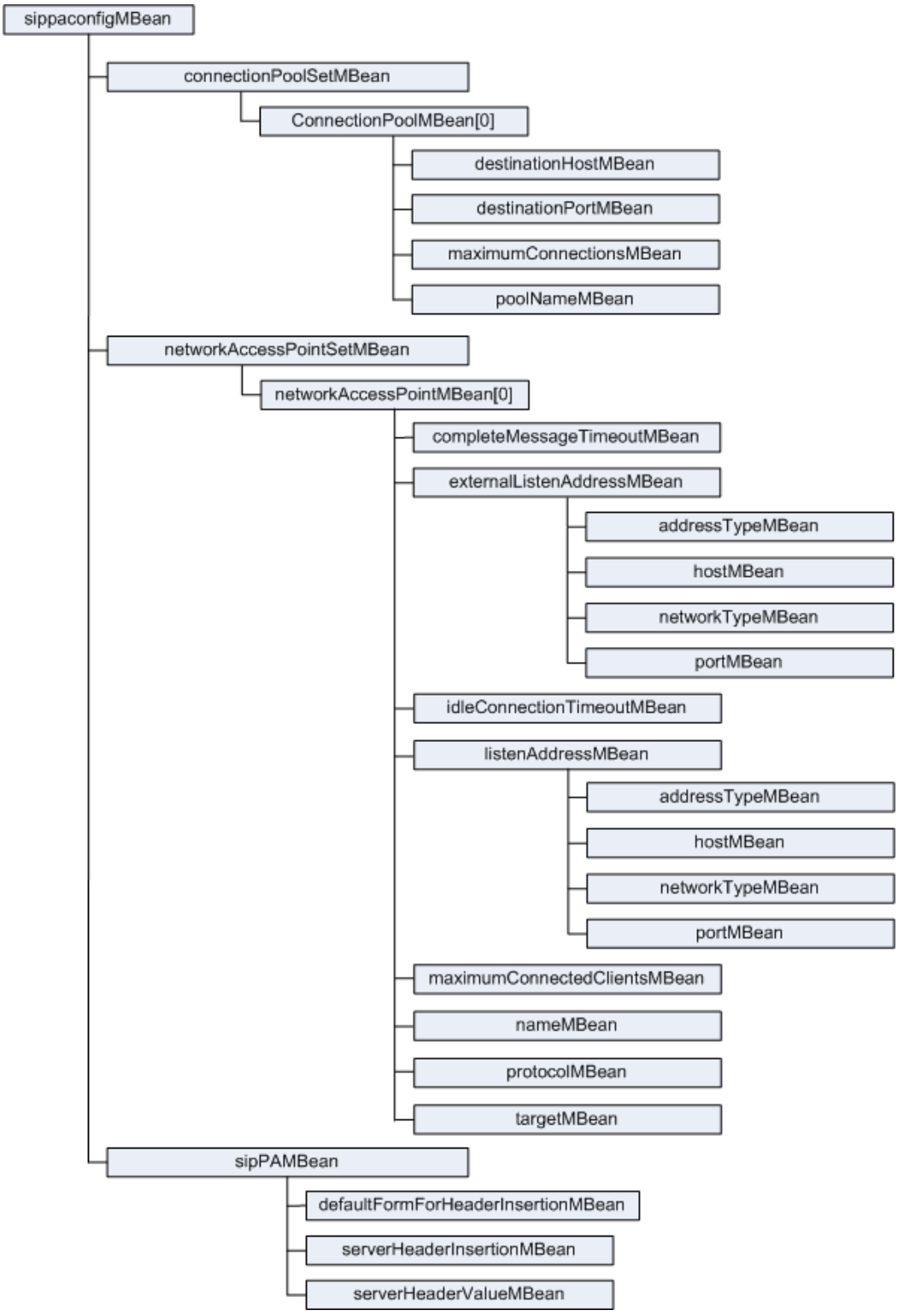
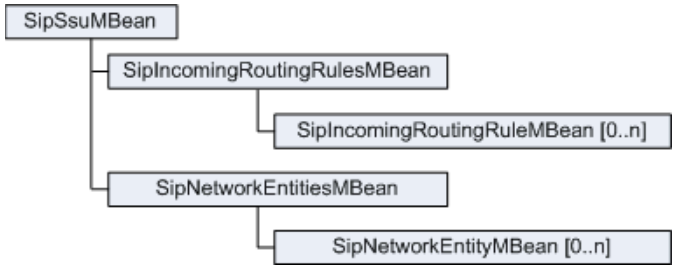


Figure 4–2 SIP SSU Configuration MBean Hierarchy



The following sections provide reference information for the SIP SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

sipPAconfigMBean

sipPAconfigMBean is the root MBean for SIP SSU configuration settings.

Factory Method

Created automatically

Attributes

- connectionPoolSet
- networkAccessPointSet
- paMessageDebug
- sipPA

Operations

None

connectionPoolSetMBean

Each instance of **ConnectionPoolSetMBean** minimizes communication overhead with SIP entities.

Factory Method

Created automatically

Attributes

None

Operations

addConnectionPool()

Adds a connection pool

removeConnectionPool()

Removes a connection pool

ConnectionPoolMBean

ConnectionPoolMBean allows you to configure multiple, fixed pools of connections to different addresses.

Factory Method

connectionPoolSet.addConnectionPool[0..n]

Attributes

None

Operations

cloneIt

Creates a copy of this MBean

destinationHostMBean

destinationHostMBean specifies the IP address or host name of the destination. Service Broker opens or re-uses connections in this pool only when making requests to the configured address.

Factory Method

Created automatically

Attributes

destinationHost

Operations

None

destinationPortMBean

destinationPortMBean specifies the port number of the destination S-CSCF for this pool.

Factory Method

Created automatically

Attributes

destinationPort

Operations

None

maximumConnectionsMBean

maximumConnectionsMBean specifies the maximum number of opened connections to maintain in this pool.

Factory Method

Created automatically

Attributes

maximumConnections

Operations

None

poolNameMBean

poolNameMBean specifies a string value that identifies the name of this pool. All configured names must be unique to the domain.

Factory Method

Created automatically

Attributes

poolName

Operations

None

networkAccessPointSetMBean

networkAccessPointSetMBean is the root MBean that defines SIP network channels that the SIP SSU uses to listen to incoming connections.

Factory Method

Created automatically

Attributes

None

Operations

addNetworkAccessPoint()

Adds an instance of NetworkAccessPointSetMBean that enables you to configure an individual SIP network access point

removeNetworkAccessPoint()

Removes an instance of NetworkAccessPointSetMBean that enables you to configure an individual SIP network access point

networkAccessPointMBean

networkAccessPointMBean defines SIP network channels that the SIP SSU uses to listen to incoming connections.

Factory Method

networkAccessPointSet.addnetworkAccessPoint[0..n]

Attributes

None

Operations

cloneIt

Creates a copy of this MBean

completeMessageTimeoutMBean

completeMessageTimeoutMBean specifies the maximum amount of time (in seconds) a network access point waits for a complete message to be received. A value of 0 disables network access point complete message timeout. Minimum value: 0, Maximum value: 480.

Factory Method

Created automatically

Attributes

completeMessageTimeout

Operations

None

externalListenAddressMBean

externalListenAddressMBean listens for the IP address or DNS name representing the external identity of this network channel. External Listen Port is the externally published listen port for a specific network access point.

Factory Method

Created automatically

Attributes

None

Operations

None

addressTypeMBean

addressTypeMBean specifies the address type. Currently, only IP4 is supported.

Factory Method

Created automatically

Attributes

addressType

Operations

None

hostMBean

hostMBean specifies the IP address or DNS name.

Factory Method

Created automatically

Attributes

host

Operations

None

networkTypeMBean

networkTypeMBean specifies the network type of the SIP network channel external identity.

Factory Method

Created automatically

Attributes

networkType

Operations

None

portMBean

portMBean specifies the address type of the SIP network external identity.

Factory Method

Created automatically

Attributes

port

Operations

None

idleConnectionTimeoutMBean

idleConnectionTimeoutMBean specifies the maximum amount of time (in seconds) that a connection can be idle before it is closed by this network access point. Minimum value is 0.

Factory Method

Created automatically

Attributes

idleConnectionTimeout

Operations

None

listenAddressMBean

listenAddressMBean Address type. Currently, only IP4 is supported.

Factory Method

Created automatically

Attributes

None

Operations

None

addressTypeMBean

addressTypeMBean specifies the address type. Currently only IP4 is supported.

Factory Method

Created automatically

Attributes

addressType

Operations

None

networkTypeMBean

networkTypeMBean specifies the network type. Currently, only internet is supported.

Factory Method

Created automatically

Attributes

networkType

Operations

None

portMBean

portMBean specifies the listen port. Minimum value: 1, Maximum value: 65535.

Factory Method

Created automatically

Attributes

port

Operations

None

maximumConnectedClientsMBean

maximumConnectedClientsMBean specifies the maximum number of clients that can be connected on this network access point. Minimum value is 1.

Factory Method

Created automatically

Attributes

maximumConnectedClients

Operations

None

nameMBean

nameMBean specifies the name of the network access point.

Factory Method

Created automatically

Attributes

name

Operations

None

protocolMBean

protocolMBean specifies the protocol the network channel should use for connections. Currently, only SIP is supported.

Factory Method

Created automatically

Attributes

protocol

Operations

None

targetMBean

targetMBean indicates which SSU this network access point is applied to. A value of null indicates that it is applied to the whole domain.

Factory Method

Created automatically

Attributes

target

Operations

None

sipPAMBean

sipPAMBean specifies that when APP and PA bundles are deployed in a domain, PA bundles share the value of the APP bundle.

Factory Method

Created automatically

Attributes

None

Operations

None

defaultFormForHeaderInsertionMBean

defaultFormForHeaderInsertionMBean specifies how the server applies rules for compacting SIP message headers. The following values are supported:

- Compact
- Force compact
- Long
- Force long

When APP and PA bundles are deployed in one domain, PA bundles share the values of the APP bundle.

Factory Method

sipPA.createdefaultFormForHeadInsertion()

Attributes

defaultFormForHeaderInsertion

Operations

None

serverHeaderInsertionMBean

serverHeaderInsertionMBean specifies the conditions for inserting a Server header into SIP messages. The following string values are supported:

- None (default)
- Request
- Response
- All

When APP and PA bundles are deployed in one domain, PA bundles share the values of the APP bundle.

Factory Method

Created automatically

Attributes

serverHeaderInsertion

Operations

None

serverHeaderValueMBean

serverHeaderInsertionMBean specifies the value of the server header inserted into SIP messages. Updating the value needs to restart the server. When APP and PA bundles are deployed in one domain, PA bundles share the values of the APP bundle

Factory Method

Created automatically

Attributes

serverHeaderValue

Operations

None

SipSsuMBean

SsuSipBean is a root MBean for the SIP connectivity configuration

Factory Method

Created automatically

Attributes

GloballyRoutableUri

For more information on this attribute, see Table 4–3.

Operations

None

SipIncomingRoutingRulesMBean

SipIncomingRoutingRulesMBean is a container for instances of SipIncomingRoutingRuleMBean. Each instance of SipIncomingRoutingRuleMBean represents an individual incoming routing rule.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSipIncomingRoutingRules()

Gets an array of references to instances of SipIncomingRoutingRuleMBean

ObjectName createSipIncomingRoutingRule()

Creates an instance of SipIncomingRoutingRuleMBean that enables you to configure an individual incoming rule

ObjectName lookupSipIncomingRoutingRule()

Returns the specified SipIncomingRoutingRuleMBean

void destroySipIncomingRoutingRule()

Destroys a SipIncomingRoutingRuleMBean

SipIncomingRoutingRuleMBean

Each instance of SipIncomingRoutingRuleMBean represents an individual incoming routing rule that defines how the SIP SSU routes incoming SIP messages to internal Service Broker IMs.

Factory Method

SipIncomingRoutingRules.createSipIncomingRoutingRule()

Attributes

- Name
- IpAddress
- Alias

For more information on each of these parameters, see Table 4–5.

Operations

None

SipNetworkEntitiesMBean

SipNetworkEntitiesMBean is a container for instances of SipNetworkEntityMBean that represent individual network entities.

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getSipNetworkEntity()

Gets an array of references to instances of SipNetworkEntityMBean

ObjectName createSipNetworkEntity()

Creates an instance of SipNetworkEntityMBean that enables you to configure an individual SIP network entity

ObjectName lookupSipNetworkEntity()

Returns a specified SipNetworkEntityMBean

void destroySipNetworkEntity()

Destroys an instance of SipNetworkEntityMBean

SipNetworkEntityMBean

Each instance of SipNetworkEntityMBean enables you to define a SIP network entity with which the SIP SSU communicate.

Factory Method

SipNetworkEntities.createSipNetworkEntity()

Attributes

- Name
- Alias
- Heartbeat
- Uri
- HeartbeatMethod
- ResponseTimeout
- ActiveInterval
- InactiveInterval

For more information on these attributes, see Table 4–1.

Operations

None

5

Configuring Diameter Signaling Server Units

This chapter describes how to configure the Service Broker Diameter SSU using the Service Broker Administration Console and Java MBeans.

About the Diameter SSU

Using the Diameter SSU, Service Broker communicates with network entities and charging applications through the Diameter protocol.

The Diameter SSU is automatically deployed and running on all Signaling Servers in the Signaling Domain.

When configuring the Diameter SSU, you set up the following:

- Incoming routes, which define how the Diameter SSU routes incoming messages to internal Service Broker components, such as Interworking Modules. See "Configuring Incoming Routes" for more information.
- Outbound destinations, which define Diameter peers to which the Diameter SSU routes outbound messages. See "Configuring Outbound Destinations" for more information.
- Diameter nodes, which define how Diameter peers access the Diameter SSU. See "Configuring Diameter Nodes" for more information.

Configuring Incoming Routes

When the Diameter SSU receives a message from the network, you need to define the internal Service Broker component to which the Diameter SSU routes the message. The routing is conditional. This means that you can set up conditions that must be met in order the message to be sent to a specific component.

To set up these conditions, you configure the following:

- Incoming Routes, defining the internal Service Broker components, such as Interworking Modules, to which the Diameter SSU routes incoming messages. See "Setting Up Incoming Routing Rules" for more information.
- Route Criteria, which define conditions for incoming routing rules. If these conditions are met, the Diameter SSU routes the incoming message to the component specified in the incoming routing rule. See "Setting Up Incoming Routing Criteria" for more information.

Setting Up Incoming Routing Rules

The Diameter SSU routes incoming Diameter requests to Service Broker components based on the value of AVPs in the request. An incoming routing rule defines a destination of Diameter requests. Criteria for routing requests to the destination, are specified in the Incoming Routing Criteria table.

The priority of the rules defines the order in which the Diameter SSU checks incoming routing rules. The Diameter SSU applies the first found rule whose criteria are met.

For example, you can create multiple rules for the same IM. The Diameter SSU begins with the rule that has the highest priority. If an incoming session fits the criteria defined in this rule, the Diameter SSU applies the rule and does not check the rest of the rules. Otherwise, the Diameter SSU checks whether an incoming session fits the criteria of a rule with a lower priority. The Diameter SSU performs this check until the Diameter SSU finds a rule whose criteria are met by an incoming session.

You can set up incoming routing rules with the Administration Console or Java MBeans.

Setting Up Incoming Routing Rules with the Administration Console

To set up incoming routing rules:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **SSU Diameter** tab.
5. Click the **Routing** subtab.
6. Click the **Incoming Routing Rules** subtab.

   The Incoming Routing Rules configuration pane appears. This pane displays a table. The table contains rules that define the component to which the Diameter SSU routes an incoming message and the priority of the rule. Each row in the table represents a single rule.
7. Click **New** at the bottom of the Incoming Routing Rules pane.

   The New dialog box appears.
8. Fill in the fields described in Table 5–1.

Table 5–1 Diameter SSU Incoming Routing Rule Fields

Name	Type	Description
Name	STRING	Specifies a unique rule name.

Table 5–1 (Cont.) Diameter SSU Incoming Routing Rule Fields

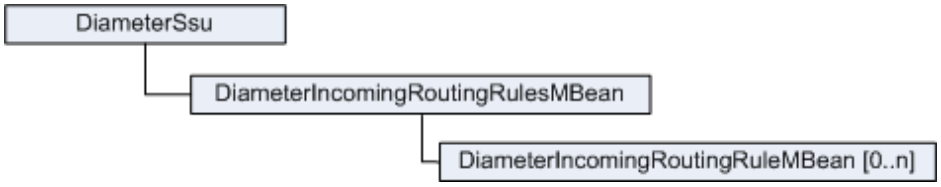
Name	Type	Description
Priority	INT	Specifies an order in which the Diameter SSU checks routing rules to evaluate if an incoming session fits rule's criteria. The Diameter SSU applies the first found rule which criteria are met by an incoming session. The lower the number, the higher the priority. For example, if you created two rules and set Priority of one rule to" 1" and set Priority of another rule to "2", the Diameter SSU checks the rule with Priority set to "1" first. You can define an incoming routing rule that the Diameter SSU applies if no other rule can be applied, by setting the Priority parameter of this rule to the largest number (that is lowest priority). There is no need to specify incoming routing criteria for such a rule.
Module Instance	STRING	Specifies the URI of the destination Service Broker component to which the Diameter SSU routes incoming sessions. The URI has the following format: SSU:*IM-instance-name.IM-type@domain-id* ■ *IM-instance-name*: The IM instance name that you specified when you added this IM in the IM Management Configuration screen. ■ *IM-type*: The type of the IM instance ■ *domain-id*: Name of the Processing Domain or Processing Domain Group where the relevant IM or application is deployed. This parameter is required only when your Service Broker deployment includes two or more Processing Domains. Use the name given to the domain when it was created. This name is specified by the axia.domain.id property. To set a Processing Domain Group, you must specify the group name. See "Managing Processing Domain Groups" in the *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information. *domain-id* is required only if your deployment includes two or more Processing Domains. For example: `ssu:imocf_instance.IMOCF@processing-domain-1`

9. Click **OK**.

Setting Up Incoming Routing Rules with Java MBeans

Figure 5–1 shows the hierarchy of the configuration MBeans that you use to configure incoming routing rules. See "Diameter SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 5–1 Diameter Incoming Routing Rules Configuration MBeans



Each incoming routing rule is represented by **DiameterIncomingRoutingRuleMBean**. You need to create a separate instance of **DiameterIncomingRoutingRuleMBean** for each incoming routing rule.

To create an incoming routing rule:

1. Create an instance of **DiameterIncomingRoutingRuleMBean** by invoking the following operation of **DiameterIncomingRoutingRulesMBean**:

   ```
   ObjectName createDiameterIncomingRoutingRule(string RuleName)
   ```

2. Set the following attributes of the newly created instance of **DiameterIncomingRoutingRuleMBean**:

   - Name
   - Priority
   - ModuleInstance

   See Table 5–1 for more information about these attributes.

Setting Up Incoming Routing Criteria

Incoming routing criteria define conditions for incoming routing rules. If these conditions are met, the Diameter SSU routes the incoming message to the IM specified in the incoming routing rule.

The conditions are based on AVPs. You can specify the attribute to be set in the incoming message and the value that this attribute must have. If these parameters match the actual attribute and its value set in the incoming message, the Diameter SSU routes this message to the IM that you defined in the appropriate incoming routing rule.

You can set up incoming routing criteria with the Administration Console and Java MBeans.

Setting Up Incoming Routing Criteria with the Administration Console

To set up incoming routing criteria:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **SSU Diameter** tab.
5. Click the **Routing** subtab.
6. Click the **Incoming Routing Criteria** subtab.

   The Incoming Routing Criteria configuration pane appears. This pane displays a table. The table contains criteria that define the conditions to be met in order the

incoming message to be sent to the Service Broker component that you defined in the incoming routing rules. Each row in the table represents a single rule.

7. Click **New** at the bottom of the Incoming Routing Criteria pane.

   The New dialog box appears.

8. Fill in the fields described in Table 5–2.

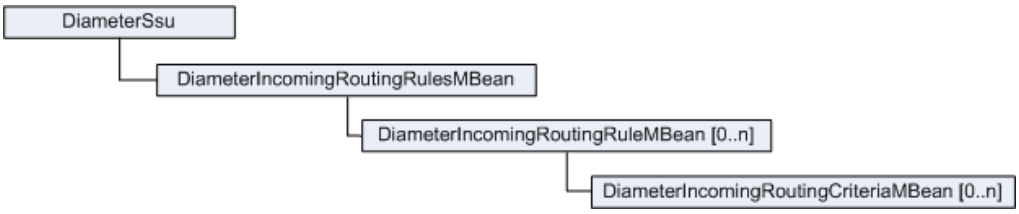
Table 5–2 Diameter SSU Incoming Routing Criteria Fields

Name	Type	Description
Parent	STRING	Specifies the name of the rule for which you define criteria. This list contains names of the rules that you created on the Incoming Routing Rules subtab.
Name	STRING	Specifies a unique rule name.
Attribute	STRING	Specifies an attribute of a Diameter AVP based on which the Diameter SSU performs routing. The Diameter SSU routes incoming messages to the specified module instance, if the value of this parameter matches the value set in the Value parameter. Possible values: ▪ APPLICATION_ID ▪ ORIGIN_REALM ▪ ORIGIN_HOST Default value: APPLICATION_ID
Value	STRING	Specifies a value that the Attribute parameter must match, in order for the rule specified in Parent to apply. You can define one of the following in the Value parameter: ▪ Single value ▪ Range of dash-separated values ▪ Comma-separated values

Setting Up Incoming Routing Criteria with Java MBeans

Figure 5–2 shows the hierarchy of the configuration MBeans that you use to configure incoming routing criteria. See "Diameter SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 5–2 Diameter Incoming Routing Criteria Configuration MBeans



Each incoming routing criterion is represented by **DiameterIncomingRoutingCriteriaMBean**. You need to create a separate instance of **DiameterIncomingRoutingCriteriaMBean** for each incoming routing criterion.

To create an incoming routing criterion:

1. Create an instance of **DiameterIncomingRoutingRuleMBean** by invoking the following operation of **DiameterIncomingRoutingRulesMBean**:

   ```
   ObjectName createDiameterIncomingRoutingRule(string RuleName)
   ```

2. Create an instance of **DiameterIncomingRoutingCriteriaMBean** by invoking the following operation of **DiameterIncomingRoutingRuleMBean**:

   ```
   ObjectName createDiameterIncomingRoutingCriteria (string CriterionName)
   ```

3. Set the following attributes of the newly created instance of **DiameterIncomingRoutingCriteriaMBean**:

   - Parent
   - Name
   - Attribute
   - Value

   See Table 5–2 for more information about these attributes.

Configuring Outbound Destinations

You can specify the destination Diameter peers to which outbound messages are routed. You can also assign an alias which is shared among multiple destination peers in the realm. The messages are dispatched based on a preconfigured strategy, according to a fixed random load share.

When configuring IMs supporting Diameter traffic, you can set a value in the Destination-Realm AVP field that matches the alias you assign to the destination peers in Outbound Destinations. The message is then redirected to the destination specified in the Outbound Destinations table.

If the value in the Destination-Realm AVP field does not match the alias you set in the Outbound Destinations table, the message is sent to the destination specified in the Destination-Host AVP field.

See "Configuring AVPs" in *Oracle Communications Server Broker Processing Domain Configuration Guide* for more information.

Configuring Outbound Destinations with the Administration Console

To add outbound destinations:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **SSU Diameter** tab.
5. Click the **Outbound Destinations** subtab.

   The Outbound Destinations configuration pane appears. This pane displays a table. The table contains rules that define the destination Diameter peer to which the Diameter SSU routes an outgoing message. Each row in the table represents a single rule.

6. Click the **New** button at the bottom of the Outbound Destinations pane.

   The New dialog box appears.

7. Fill in the fields described in Table 5–3.

Table 5–3 Diameter SSU Outbound Destinations Parameters

Field	Descriptions
Name	Specifies a unique destination identifier.
Alias	Specifies the alias you assign to the destination Diameter peer. You can assign the same alias to a number of Diameter peers in the realm to dispatch messages to multiple nodes in the realm.
Destination Host	Specifies the host of the destination Diameter peer.
Destination Realm	Specifies the realm to which the destination Diameter peer belongs.

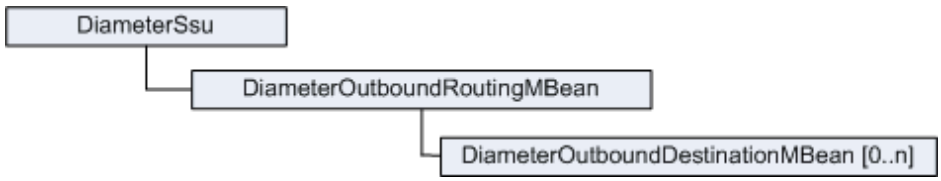
8. Click **OK**.

   The Diameter SSU dispatches messages to destination Diameter peers in the realm according to a preconfigured strategy.

Configuring Outbound Destinations with JavaMBeans

Figure 5–3 shows the hierarchy of the configuration MBeans that you use to configure outbound destinations. See "Diameter SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 5–3 Diameter Outbound Destination Configuration MBeans



Each destination is represented by **DiameterOutboundDestinationMBean**. You need to create a separate instance of **DiameterOutboundDestinationMBean** for each destination.

To create an outbound destination:

1. Create an instance of **DiameterOutboundDestinationMBean** by invoking the following operation of **DiameterOutboundRoutingMBean**:

   ```
   ObjectName createDiameterOutboundDestination(string RuleName)
   ```

2. Set the following attributes of the newly created instance of **DiameterIncomingRoutingRuleMBean**:

   - Name
   - Alias
   - DestinationHost
   - DestinationRealm

   See Table 5–3 for more information about these attributes.

Configuring Diameter Nodes

The Diameter SSU is a process that implements the Diameter protocol. You define the Diameter SSU as a Diameter node and configure how other Diameter entities access the Diameter node.

By default, the Diameter SSU is configured as one Diameter node, where all Signaling Servers provide a Diameter network channel on the same port. If you deploy the Diameter SSU on multiple Signaling Servers running on the same physical machine, you must configure each Signaling Server to listen on a different port, otherwise the ports will collide. In this case, you define a Diameter node for each Signaling Server. In general, when you need to define a different address, host or port to Diameter SSU deployments running on different Signaling Servers, you create a Diameter node for each Signaling Server.

You can perform the following actions with Diameter nodes:

- Creating a new node. See "Creating a Diameter Node" for more information.
- Setting up the default route. See "Configuring the Default Route" for more information.
- Setting up routes. See "Configuring Routes" for more information.
- Setting up peers. See "Configuring Peers" for more information.

Creating a Diameter Node

You can create a new Diameter node with the Administration Console or Java MBeans.

Creating a Diameter Node with the Administration Console

To create a Diameter node:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **DIAMETER** tab.

   The Diameter configuration pane appears. This pane contains the tools that you need to configure Diameter nodes.
5. In the **Node** area, click the **Plus** icon.

   The fields of the General tab appear.
6. Fill in the fields described in Table 5–4.

Table 5–4 Diameter Node Parameters

Field	Description
Name	Specifies a unique name for the Diameter configuration.
Realm	Specifies the realm name of the Diameter node. For example: **host@oracle.com** Multiple Diameter nodes can be run on a single host using different realms and listen port numbers.

Table 5–4 (Cont.) Diameter Node Parameters

Field	Description
Address	Specifies the listen address for this Diameter node, using either the DNS name or the IP address. The host identity is used as the listen address when this field is blank. The host identity might or might not match the DNS name. Oracle recommends configuring the **Address** property with an explicit DNS name or IP address to avoid configuration errors.
Host	Specifies the host name of the Diameter node. The host identity might or might not match the DNS name.
Port	Specifies the network port number to use with the listen address.
Target	Specifies the name of a single Signaling Server (JVM) that provides the Diameter network channel. Leave the field blank to specify that all the Signaling Servers in the domain provide Diameter network channels. There can be only one blank entry for the **Target** field, which makes it the default Signaling Server. The Target field includes the following options: ■ Include Origin State ID: Specifies that the Origin State ID AVP is included in each request, which allows for the rapid detection of terminated sessions. Diameter AVPs carry specific authentication, accounting, authorization routing and security information, as well as configuration details for request and reply. ■ SCTP: Indicates that the Diameter node is configured with support for SCTP. ■ TLS: Indicates that the Diameter node is configured with support of Transport Layer Security (TLS). This field is used to advertise TLS capabilities when the node is interrogated by another Diameter node.
Request Timeout	Specifies the amount of time, from 0 milliseconds, this node waits for an answer message before timing out.
Watchdog Timeout	Specifies the amount of time, from 0 seconds, this node uses for the value of the Diameter Tw watchdog timer interval.

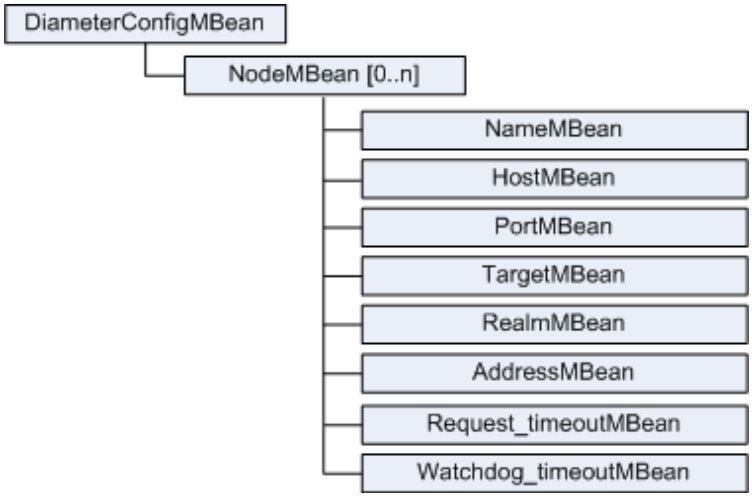
After you have created a general node configuration, the configuration name appears in the list of Diameter nodes. You can select the node to configure Diameter routes and peers as described in the following sections:

- Configuring the Default Route with the Administration Console
- Configuring Routes with the Administration Console
- Configuring Peer with the Administration Console

Creating a Diameter Node with Java MBeans

Figure 5–4 shows the hierarchy of the configuration MBeans that you use to create a Diameter node. See "Diameter SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 5–4 Diameter Node Configuration MBeans



Each Diameter node is represented by an instance of **NodeMBean**. You need to create a separate instance of **NodeMBean** for each Diameter node. When you create an instance of **NodeMBean**, the set of child MBeans are created. Each child MBean represent a single parameter of the Diameter node.

For example, **RealmMBean** has the Realm attribute that represents the Realm parameter.

To create a Diameter node:

1. Create an instance of **NodeMBean** by invoking the following operation of **DiameterConfigMBean**:

   ```
   void addNode()
   ```

   NodeMBean is created with the following child MBeans:

   - **addressMBean**
   - **nameMBean**
   - **realmMBean**
   - **message-debug-enabledMBean**

2. Create additional child MBeans of **NodeMBean** by invoking the following operations of **NodeMBean**:

   ```
   void addHost()
   void addPort()
   void addTarget()
   void addRequest_timeout()
   void addWatchdog_timeout()
   ```

3. Set the attributes of the child MBeans as described in Table 5–5.

Table 5–5 NodeMBean Child MBeans and Their Attributes

MBean	Attribute
nameMBean	Name
realmMBean	Realm
addressMBean	Address

Table 5–5 (Cont.) NodeMBean Child MBeans and Their Attributes

MBean	Attribute
hostMBean	Host
portMBean	Port
targetMBean	Target
request_timeoutMBean	Request_timeout
watchdog_timeoutMBean	Watchdog_timeout

See Table 5–4 for more information about these attributes.

Configuring the Default Route

You can specify the default route with the Administration Console or Java MBeans.

Configuring the Default Route with the Administration Console

To configure the default route:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **DIAMETER** tab.
5. In the **Node** area, select the node for which you set up the default route.
6. Click the **Default Route** subtab.
7. Fill in the fields described in Table 5–6.

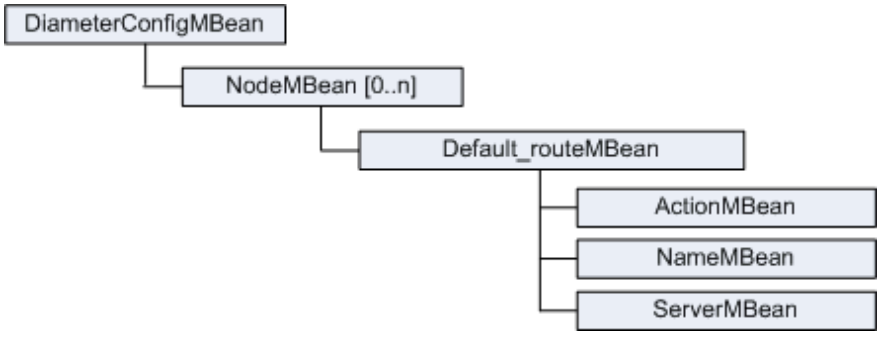
Table 5–6 Diameter Default Route Parameters

Field	Description
Name	Specifies an administrative name for the route.
Default Route Action	Specifies an action that this node performs when using the default route. The action type can be any of the following: ■ None ■ Local ■ Relay ■ Proxy ■ Redirect
Server	Clicking the **New** button (+) at the bottom of the Default Route pane opens the New Data dialog box where you specify the name of the target peers that will use the route. The target server maps to the Host field in the Peer panel. See "Configuring Peer with the Administration Console" for more information.

Configuring the Default Route with Java MBeans

Figure 5–5 shows the hierarchy of the configuration MBeans that you use to configure the default route of a Diameter node. See "Diameter SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 5–5 Diameter Default Route Configuration MBeans



Each default route is represented by an instance of **default-routeMBean**. When you create an instance of **NodeMBean**, the set of child MBeans are created. Each child MBean represent a single parameter of the default route.

For example, **actionMBean** has the Action attribute that represents the Action parameter.

To set up the default route of a Diameter node:

1. Create an instance of **default-routeMBean** by invoking the following operation of **NodeMBean**:

   ```
   void addDefault_route()
   ```

   default-routeMBean is created with the following child MBeans:

   - **nameMBean**
   - **actionMBean**
   - **serverMBean**

2. Set the attributes of these child MBeans as described in Table 5–7.

Table 5–7 default-routeMBean Child MBeans and Their Attributes

MBean	Attribute
nameMBean	Name
actionMBean	Realm
serverMBean	Address

See Table 5–6 for more information about these attributes.

Configuring Routes

Certain Diameter nodes, such as relays, configure realm-based routes for use when resolving Diameter messages.

Configuring Routes with the Administration Console

To configure a new Diameter route:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.

4. In the SSU Diameter configuration pane, click the **DIAMETER** tab.
5. In the **Node** area, select the node for which you set up the routes.
6. Click the **Routes** subtab.
7. Fill in the fields described in Table 5–8.

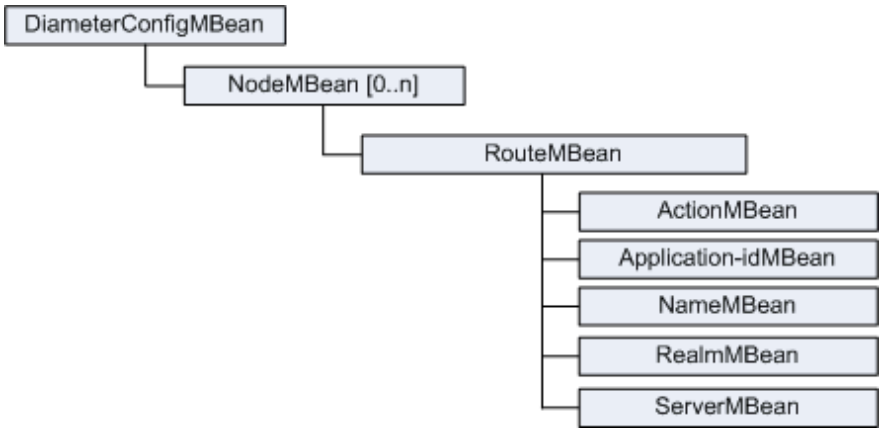
Table 5–8 Routes Parameters

Field	Description
Name	Specifies an administrative name for the route.
Action	Specifies an action that this node performs when using the route. Possible values: ■ None ■ Local ■ Relay ■ Proxy ■ Redirect
Realm	Specifies the target realm for this route.
Application ID	Specifies the type of Diameter billing to use. Possible values ■ **3** Specifies Diameter Rf charging. ■ **4** Specifies Diameter Ro charging.
Server	Clicking the New button (+) at the bottom of the Default Route pane opens the New Data dialog box where you specify the name of the target peers that will use the route. The target server maps to the Host field in the Peer pane. See "Configuring Peer with the Administration Console" for more information.

Configuring Routes with Java MBeans

Figure 5–6 shows the hierarchy of the configuration MBeans that you use to configure routes of a Diameter node. See "Diameter SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 5–6 Diameter Route Configuration MBeans



Each route is represented by an instance of **routeMBean**. You need to create a separate instance of **routeMBean** for each route of the Diameter node. When you create an instance of **routeMBean**, the set of child MBeans are created. Each child MBean represent a single parameter of the route.

For example, **actionMBean** has the Action attribute that represents the Action parameter.

To set up the default route of a Diameter node:

1. Create an instance of **default-routeMBean** by invoking the following operation of **NodeMBean**:

   ```
   void addRoute()
   ```

   routeMBean is created with the following child MBeans:

   - **nameMBean**
   - **actionMBean**
   - **serverMBean**
   - **realmMBean**
   - **application-idMBean**

2. Set the attributes of these child MBeans as described in Table 5–9.

Table 5–9 routeMBean Child MBeans and Their Attributes

MBean	Attribute
nameMBean	Name
actionMBean	Realm
serverMBean	Address
realmMBean	Realm
application-idMBean	Application-id

See Table 5–8 for more information about these attributes.

Configuring Peers

A Diameter node defines peer connection information for each Diameter node in the realm, or enables dynamic peers in combination with TLS transport to allow peers to be recognized automatically.

Configuring Peer with the Administration Console

To configure a new Diameter peer node:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU Diameter** node.
4. In the SSU Diameter configuration pane, click the **DIAMETER** tab.
5. In the **Node** area, select the node for which you set up the peers.
6. Click the **Peers** subtab.
7. In the Peers configuration pane, define the parameters described in Table 5–10.

Table 5–10 Peer Recognition Parameters

Field	Description
Allow Dynamic Peers	Enables dynamic discovery of Diameter peers. Dynamic peer support is disabled by default. Oracle recommends enabling dynamic peers only when using the TLS transport, because no access control mechanism is available to restrict hosts from becoming peers. If you enabled dynamic peers, you can set two additional parameters: ▪ **diameter.watchdog.for.dynamic.peers** This parameter defines whether or not the Diameter SSU should send Device-Watchdog-Request (DWR) commands to dynamic Diameter peers. ▪ **diameter.tcp.keepalive.for.client.peers** This parameter defines whether or not the TCP socket option SO_KEEPALIVE for Diameter dynamic peers is set to true. You define these parameters in the **start.sh** file of the Signaling Server on which the Diameter SSU runs. See the "System Properties" section in the "System Administrator's Reference" chapter of the *Oracle Communications Service Broker System Administrator's Guide*.
Peer Retry Delay	Specifies the time, in seconds. This node waits before retrying a request to a Diameter peer. The default wait value is 30 seconds.

8. In the **Peers** configuration pane, click the **New** button (+) to define a new peer entry.

   The New Data dialog box appears.
9. Fill in the fields described in Table 5–11.

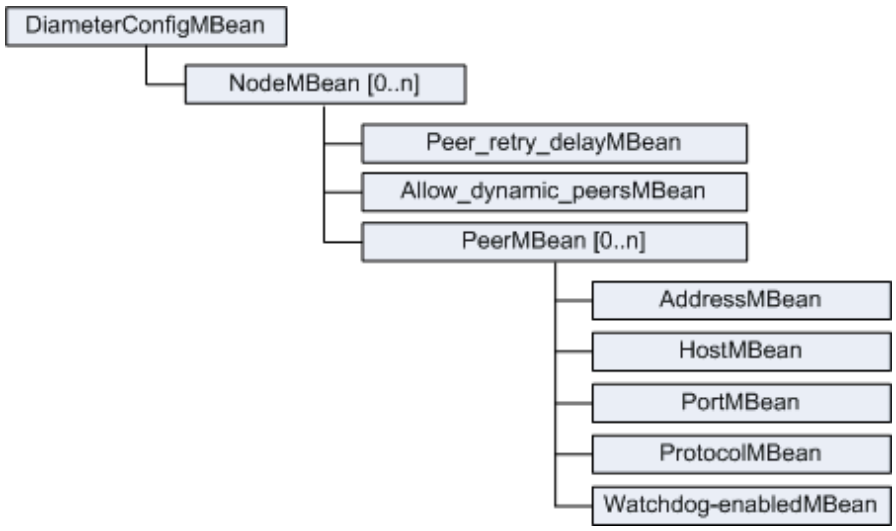
Table 5–11 Peer Recognition Parameters

Field	Description
Address	Specifies the peer's address, using either the DNS name or IP address.
Host	Specifies the peer's host identity.
Port	Specifies the listen port number of the peer.
Protocol	Specifies the protocol used to communicate with the peer. Possible values: ▪ tcp ▪ sctp Default value: tcp Note that Service Broker attempts to connect to the peer using *only* the protocol you specify. The other protocol is not used, even if a connection fails using the selected protocol.
Watchdog	Indicates whether the peer supports the Diameter Tw watchdog timer interval.

Configuring Peers with Java MBeans

Figure 5–7 shows the hierarchy of the configuration MBeans that you use to configure peers. See "Diameter SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 5–7 Diameter Peers Configuration MBeans



Each peer is represented by an instance of **peerMBean**. You need to create a separate instance of **peerMBean** for each peer. When you create an instance of **peerMBean**, the set of child MBeans are created. Each child MBean represent a single parameter of the route.

For example, **portMBean** has the Port attribute that represents the **Port** parameter.

To set up a peer of a Diameter node:

1. Create an instance of **peerMBean** by invoking the following operation of **NodeMBean**:

```
void addPeer()
```

peerMBean is created with the following child MBeans:

- **addressMBean**
- **hostMBean**
- **portMBean**
- **protocolMBean**
- **watchdog_enabledMBean**

2. Set the attributes of these child MBeans as described in Table 5–12.

Table 5–12 peerMBean Child MBeans and Their Attributes

MBean	Attribute
addressMBean	Address
hostMBean	Host
portMBean	Port
protocolMBean	Protocol
watchdog_enabledMBean	Watchdog_enabled

See Table 5–10 for more information about these attributes.

3. Invoke the following operations of **NodeMBean**:

```
void addAllow_dynamic_peers()
void addPeer_retry_delay()
```

These operations created **allow_dynamic_peersMBean** and **peer_retry_delayMBean** accordingly.

4. Set the allow-dynamic-peers attribute of **allow_dynamic_peersMBean** as described in Table 5–11.

5. Set the peer_retry_delay attribute of **peer_retry_delayMBean** as described in Table 5–11.

Diameter SSU Configuration MBeans Reference

The following sections provide reference information for the Diameter SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

DiameterIncomingRoutingRulesMBean

DiameterIncomingRoutingRulesMBean enables you to manage incoming routing rules. Each incoming routing rule is represented by am instance of DiameterIncomingRoutingRuleMBean. Using DiameterIncomingRoutingRulesMBean, you need to create a separate instance of DiameterIncomingRoutingRuleMBean for each rule.

Object Name

com.convergin:Type=DiameterIncomingRoutingRules,Version=*MBean_Version*,Location=AdminServer,Name=ssudiameter.diameter_inbound_routing

Factory Method

Created automatically

Attributes

None

Operations

ObjectName[] getDiameterIncomingRoutingRules()

Gets an array of references to instances of DiameterIncomingRoutingRuleMBean

ObjectName createDiameterIncomingRoutingRule()

Creates an instance of DiameterIncomingRoutingRuleMBean that enables you to configure an individual rule

ObjectName lookupDiameterIncomingRoutingRule()

Returns the specified DiameterIncomingRoutingRuleMBean

void destroyDiameterIncomingRoutingRule()

Destroys a DiameterIncomingRoutingRuleMBean

DiameterIncomingRoutingRuleMBean

Each instance of DiameterIncomingRoutingRuleMBean represents an individual incoming routing rule that defines an internal Service Broker component to which the Diameter SSU routes the incoming Diameter message and the priority of the rule. You need to create a separate instance of DiameterIncomingRoutingRuleMBean for each rule.

Object Name

com.convergin:Type=DiameterIncomingRoutingRule,Version=*MBean_Version*,Location=AdminServer,Name=ssudiameter.*Instance_Name*

Factory Method

DiameterIncomingRoutingRules.createDiameterIncomingRoutingRule()

Attributes

- Name
- Priority
- ModuleInstance

For more information on each of these parameters, see Table 5–13.

Operations

ObjectName createDiameterIncomingRoutingCriteria()

Creates an instance of DiameterIncomingRoutingCriteriaMBean

void destroyDiameterIncomingRoutingCriteria()

Destroys an instance of DiameterIncomingRoutingCriteriaMBean

ObjectName lookupDiameterIncomingRoutingCriteria()

Returns the reference to a specified instance of DiameterIncomingRoutingCriteriaMBean

DiameterIncomingRoutingCriteriaMBean

Each instance of DiameterIncomingRoutingCriteriaMBean represents an individual incoming routing criterion that defines the conditions to be met in order the incoming message to be routed to the internal Service Broker component specified in the incoming routing rule.

Object Name

com.convergin:Type=DiameterIncomingRoutingCriteria,Version=*MBean_Version*,Location=AdminServer,Name=ssudiameter.*Instance_Name*

Factory Method

DiameterIncomingRoutingRule.createDiameterIncomingRoutingCriteria()

Attributes

- Attribute
- Value

For more information on each of these parameters, see Table 5–2.

Operations

None

DiameterOutboundRoutingMBean

DiameterOutboundRoutingMBean enables you to manage outbound routing rules. Each outbound routing rule is represented by an instance of DiameterOutboundDestinationMBean. Using DiameterOutboundRoutingMBean, you need to create a separate instance of DiameterOutboundDestinationMBean for each outbound routing rule.

Object Name

com.convergin:Type=DiameterOutboundRouting,Version=*MBean_Version*,Location=AdminServer,Name=ssudiameter.diameter_outbound_routing

Factory Method

Created automatically

Attributes

None

Operations

createDiameterOutboundDestination()

Creates an instance of DiameterOutboundDestinationMBean that enables you to configure an outbound destination

destroyDiameterOutboundDestination()

Destroys an instance of DiameterOutboundDestinationMBean

lookupDiameterOutboundDestination()

Searches for instances of DiameterOutboundDestinationMBean

DiameterOutboundDestinationMBean

Each instance of DiameterOutboundRoutingMBean represents an outbound destination that defines how the Diameter SSU routes Diameter messages to a destination peer.

Object Name

com.convergin:Type=DiameterOutboundDestination,Version=*MBean_Version*,Location=AdminServer,Name=ssudiameter.*Instance_Name*

Factory Method

DiameterOutboundRouting.createDiameterOutboundDestination()

Attributes

- Name
- Alias
- DestinationHost
- DestinationRealm

For more information on each of these parameters, see Table 5–3.

Operations

None

DiameterConfigMBean

DiameterConfigMBean is a root MBean for this Diameter SSU configuration.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig

Factory Method

Created automatically

Attributes

Table 5–13 DiameterConfigMBean Attribute

Name	Type	Description
Name	STRING	Specifies a name for the Diameter configuration

Operations

void addName()

Adds a new instance of NameMBean that enables you to define the name for a Diameter configuration.

void removeName()

Removes an existing instance of NameMBean that defines the name of a Diameter configuration.

addNode()

Adds a new instance of NodeMBean represents all Diameter SSUs as a Diameter node.

removeNode(int NodeIndex)

Removes a specified instance of a NodeMBean.

NameMBean

NameMBean defines the name of a Diameter configuration.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=name

Factory Method

diameterConfig.addName()

Attributes

Table 5–14 NameMBean Attribute

Name	Type	Description
Name	STRING	Name of the Diameter configuration

Operations

None

NodeMBean

NodeMBean represents a Diameter SSU as a Diameter node.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0]

Factory Method

diameterConfig.addNode()

Attributes

None

Operations

void cloneIt()

Adds a copy of the NodeMBean instance that invoked this operation.

void addName()

Adds a new instance of NameMBean that defines the name of the Diameter peer.

void removeName()

Removes an instance of NameMBean.

void addTarget()

Adds a new instance of TargetMBean that defines the ID of the Signaling Server for the Diameter node.

void removeTarget()

Removes an instance of TargetMBean.

void addHost()

Adds a new instance of HostMBean that defines the host identity of the Diameter node.

void removeHost()

Removes an instance of HostMBean.

void addRealm()

Adds a new instance of RealmMBean that defines the name of a realm for which the Diameter node is responsible.

void removeRealm()

Removes an instance of RealmMBean.

void addAddress()

Adds a new instance of AddressMBean that defines the listen address for the Diameter node using either the DNS name or IP address.

void removeAddress()

Removes an instance of AddressMBean.

void addPort()

Adds a new instance of PortMBean that defines the listen port number for the Diameter node.

void removePort()

Removes an instance of PortMBean.

void addTls_enabled()

Adds a new instance of Tls_enabledMBean that defines whether or not the Transport Layer Security (TLS) is enabled.

void removeTls_enabled()

Removes an instance of Tls_enabledMBean.

void addSctp_enabled()

Adds a new instance of Sctp_enabledMBean that defines whether or not the Stream-Control TransmissionProtocol (SCTP) is enabled.

void removeSctp_enabled()

Removes an instance of Sctp_enabledMBean.

void addDebug_enabled()

Adds a new instance of Debug_enabledMBean that defines whether or not generation of debug message is enabled.

void removeDebug_enabled()

Removes an instance of Debug_enabledMBean.

void addMessageDebug_enabled()

Adds a new instance of MessageDebug_enabledMBean that defines whether or not tracing for Diameter messages processed by the Diameter node is enabled.

void removeMessageDebug_enabled()

Removes an instance of MessageDebug_enabledMBean.

void addMessage_debug()

Adds a new instance of Message_debugMBean that writes Diameter events to the log.

void removeMessage_debug()

Removes an instance of Message_debugMBean.

void addPeer_retry_delay()

Adds a new instance of Peer_retry_delayMBean that defines period of time, in seconds, between retries to connect to the peer.

void removePeer_retry_delay()

Removes an instance of Peer_retry_delayMBean.

void addAllow_dynamic_peers()

Adds a new instance of Allow_dynamic_peersMBean that defines whether or not dynamic peers are allowed.

void removeAllow_dynamic_peers()

Removes an instance of Allow_dynamic_peersMBean.

void addRequest_timeout()

Adds a new instance of Request_timeoutMBean that defines period of time, in milliseconds, that the Diameter node has to wait for an answer from the peer before timing out.

void removeRequest_timeout()

Removes an instance of Request_timeoutMBean.

void addWatchdog_timeout()

Adds a new instance of Watchdog_timeoutMBean that defines period of time, in seconds, for the Diameter Tw watchdog timer.

void removeWatchdog_timeout()

Removes an instance of Watchdog_timeoutMBean.

void addInclude_origin_state_id()

Adds a new instance of Include_origin_state_idMBean that defines whether the Origin-State-Id attribute-value pair (AVP) is included in each request.

void removeInclude_origin_state_id()

Removes an instance of Include_origin_state_idMBean.

void addPeer()

Adds a new instance of PeerMBean that defines a Diameter peer in the network with which the Diameter node communicates.

void removePeer()

Removes an instance of PeerMBean.

void addDefaultRoute()

Adds a new instance of DefaultRouteMBean that defines a route to use when a request does not match a configured route.

void removeDefaultRoute()

Removes an instance of DefaultRouteMBean.

HostMBean

HostMBean defines the host identity of this Diameter node.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=host

Factory Method

Node.addHost()

Attributes

Table 5–15 HostMBean Attribute

Name	Type	Description
Host	String	Specifies the host name of the Diameter node

Operations

None

PortMBean

PortMBean defines the listen port for a Diameter node.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=port

Factory Method

Node.addPort()

Attributes

Table 5–16 PortMBean Attribute

Name	Type	Description
Port	STRING	Specifies the listen port number for this node

Operations

None

TargetMBean

TargetMBean defines the ID of the Signaling Server for this node, as defined in the domain configuration.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=target

Factory Method

Node.addTarget()

Attributes

Table 5–17 TargetMBean Attribute

Name	Type	Description
Target	STRING	Specifies the ID of the Signaling Server for this node, as defined in the domain configuration.

Operations

None

RealmMBean

RealmMBean defines the name of a realm for which this node is responsible. An HSS, Application Server, and relay agents must all agree on a realm name or names. The realm name for the HSS and Application Server do not have to match.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=realm

Factory Method

Node.addRealm()

Attributes

Table 5–18 RealmMBean Attribute

Name	Type	Description
Realm	STRING	Specifies the realm name of the Diameter node

Operations

None

AddressMBean

AddressMBean defines the listen address for a Diameter node using either the DNS name or IP address.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=address

Factory Method

Node.addAddress()

Attributes

Table 5–19 AddressMBean Attribute

Name	Type	Description
Address	STRING	Specifies the listen address for this Diameter node using either the DNS name or IP address

Operations

None

Peer_retry_delayMBean

Peer_retry_delayMBean defines period of time, in seconds, between retries to connect to the peer.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=peer-retry-delay

Factory Method

Node.addPeer_retry_delay()

Attributes

Table 5–20 Peer_retry_delayMBean Attribute

Name	Type	Description
Peer_retry_delay	STRING	Specifies period of time, in seconds, between retries to connect to the peer

Operations

None

Allow_dynamic_peersMBean

Allow_dynamic_peersMBean defines whether or not dynamic peers are allowed.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=allow-dynamic-peers

Factory Method

Node.addAllow_dynamic_peers()

Attributes

Table 5–21 Allow_dynamic_peersMBean Attribute

Name	Type	Description
Allow_dynamic_peers	BOOL	Specifies whether or not dynamic peers are allowed. Possible values: ■ True ■ False Default value: False **Note**: It is recommended to enable dynamic peers only when using the TLS transport because no access control mechanism is available to restrict hosts from becoming peers.

Operations

None

Request_timeoutMBean

Request_timeoutMBean defines period of time, in milliseconds, that the node has to wait for an answer before timing out.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=request-timeout

Factory Method

Node.addRequest_timeout()

Attributes

Table 5–22 Request_timeoutMBean Attribute

Name	Type	Description
Request_timeout	STRING	Specifies period of time, in milliseconds, that the node has to wait for an answer before timing out

Operations

None

Watchdog_timeoutMBean

Watchdog_timeoutMBean defines period of time, in seconds, for the Diameter Tw watchdog timer.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=watchdog-timeout

Factory Method

Node.addWatchdog_timeout()

Attributes

Table 5–23 Watchdog_timeoutMBean Attribute

Name	Type	Description
Watchdog_timeout	STRING	Specifies period of time, in seconds, for the Diameter Tw watchdog timer

Operations

None

PeerMBean

PeerMBean defines a Diameter peer in the network which the Signaling Server communicate with. A Diameter node (NodeMBean) defines peer connection for each other Diameter node.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0]

Factory Method

Node.addPeer()

Attributes

None

Operations

void cloneIt()

Adds a copy of the PeerMBean instance that invoked this operation

void addHost()

Adds a new instance of HostMBean that defines the peer's host identity

void removeHost()

Removes an instance of HostMBean

void addAddress()

Adds a new instance of AddressMBean that defines the peer's listen address

void removeAddress()

Removes an instance of AddressMBean

void addPort()

Adds a new instance of PortMBean that defines the listen port number of the peer

void removePort()

Removes an instance of PortMBean

void addProtocol()

Adds a new instance of ProtocolMBean that defines the protocol used to communicate with the peer

void removeProtocol()

Removes an instance of ProtocolMBean

void addWatchdog-enabled()

Adds a new instance of Watchdog-enabledMBean that defines whether or not the peer supports the Diameter Tw watchdog timer interval

void removeWatchdog-enabled()

Removes an instance of Watchdog-enabledMBean

AddressMBean

AddressMBean defines the peer's listen address.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0], name3=address

Factory Method

Peer.addAddress()

Attributes

Table 5–24 AddressMBean Attribute

Name	Type	Description
Address	STRING	Specifies the peer's listen address (DNS name or IP address)

Operations

None

HostMBean

HostMBean defines the peer node's host identity

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0], name3=host

Factory Method

Peer.addHost()

Attributes

Table 5–25 HostMBean Attribute

Name	Type	Description
Host	STRING	Specifies the peer's host identity

Operations

None

PortMBean

PortMBean defines the listen port number of the peer

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0], name3=port

Factory Method

Peer.addPort()

Attributes

Table 5–26 PortMBean Attribute

Name	Type	Description
Port	STRING	Specifies the listen port number of the peer

Operations

None

ProtocolMBean

ProtocoMBean defines the protocol used to communicate with the peer.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0], name3=protocol

Factory Method

Peer.addProtocol()

Attributes

Table 5–27 ProtocolMBean Attribute

Name	Type	Description
Protocol	STRING	Specifies the protocol used to communicate with the peer. Possible values: ▪ tcp ▪ sctp Default value: tcp Note that Service Broker attempts to connect to the peer using *only* the protocol you specify. The other protocol is not used, even if a connection fails using the selected protocol.

Operations

None

Watchdog-enabledMBean

Watchdog-enabledMBean defines whether or not the peer supports the Diameter Tw watchdog timer interval

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=peer[0], name3=watchdog-enabled

Factory Method

Peer.addWatchdog-enabled()

Attributes

Table 5–28 Watchdog-enabledMBean Attribute

Name	Type	Description
WatchdogEnabled	BOOL	Specifies whether or not the peer supports the Diameter Tw watchdog timer interval. Possible values: ■ True ■ False Default value: False

Operations

None

RouteMBean

RouteMBean defines a server to which an IM-OCF forwards a Diameter message with the specified host realm and application ID.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=route[0]

Factory Method

Node.createRoute()

Attributes

None

Operations

void addName()

Adds an instance of NameMBean that defines the name of the default route.

void removeName()

Removes an instance of NameMBean.

void addAction()

Adds an instance of ActionMBean that defines an action that the node performs when using the configured route.

void removeAction()

Removes an instance of ActionMBean.

void addServer ()

Adds an instance of ServerMBean that defines a server to which Service Broker routes a message.

void removeServer(int ServerIndex)

Remove an instance of ServerMBean.

void addApplication_id()

Adds an instance of Application-idMBean that defines a Diameter interface used.

void removeApplication_id()

Remove an instance of Application-idMBean.

void addRealm()

Adds an instance of RealmMBean that defines the name of a realm for which this node is responsible.

void removeRealm()

Remove an instance of RealmMBean.

Application-idMBean

Application-idMBean defined a Diameter interface used.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=route[0], name3=application-id

Factory Method

Node.createRoute()

Attributes

Table 5–29 Application-idMBean Attribute

Name	Type	Description
Application-id	STRING	Specifies a Diameter interface. Possible values: ■ 4 Stands for the Ro interface ■ 16777217 Stands for the Sh interface

Operations

None

Default-routeMBean

Default-routeMBean defines a route to use when a request does not match a configured route.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=default-route

Factory Method

Created automatically

Attributes

None

Operations

void addName()

Adds an instance of NameMBean that defines the name of the default route.

void removeName()

Removes an instance of NameMBean.

void addAction()

Adds an instance of ActionMBean that defines an action that the node performs when using the configured route.

void removeAction()

Removes an instance of ActionMBean.

void addServer ()

Adds an instance of ServerMBean that defines a server to which Service Broker routes a message.

void removeServer(int ServerIndex)

Remove an instance of ServerMBean.

NameMBean

NameMBean defines the name of the default route.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0],name2=default-route, name3=name

Factory Method

Default-route.addName()

Attributes

Table 5–30 NameMBean Attribute

Name	Type	Description
Name	STRING	Specifies the name of the default route.

Operations

None

ActionMBean

ActionMBean defines an action that the node performs when using the configured route.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=default-route,name3=action

Factory Method

Default-route.addAction()

Attributes

Table 5–31 ActionMBean Attribute

Name	Type	Description
Action	STRING	Specifies an action that the node performs when using the configured route. Possible values: ■ Local ■ Relay ■ Proxy ■ Redirect Possible values: Local

Operations

None

ServerMBean

ServerMBean defines a server to which Service Broker routes a message.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter, version=*MBean_Version*,name0=diameterConfig,name1=node[0], name2=default-route,name3=server[0]

Factory Method

Created automatically

Attributes

Table 5–32 ServerMBean Attribute

Name	Type	Description
Server	STRING	Specifies a server.

Operations

CloneIt

6

Configuring RADIUS Signaling Server Units

The following sections describe how to configure Service Broker RADIUS SSUs using the Service Broker Administration Console and Java MBeans.

About the RADIUS SSU

Using the RADIUS SSU, Service Broker receives RADIUS accounting and access requests from the network and forwards these requests to an appropriate application.

When configuring the RADIUS SSU, you set up the following:

- Incoming routing rules, which define the IM to which the RADIUS SSU forwards accounting and access RADIUS requests. See "Configuring Incoming Routing Rules" for more information.
- Providing a custom dictionary, which define vendor-specific AVPs. See "Providing a Custom Dictionary" for more information.
- Server parameters, which define how the RADIUS SSU receives RADIUS requests. See "Configuring Server Parameters" for more information.
- RADIUS clients, which define the RADIUS clients from which the RADIUS SSU receives requests. See "Setting Up RADIUS Clients" for more information.
- Proxy realms, which define the realm and servers to which the RADIUS SSU routes incoming RADIUS requests when you do not want to use the local realm. See "Configuring Proxy Realm" for more information.

Configuring Incoming Routing Rules

Incoming routing rules define the IM to which the RADIUS SSU routes RADIUS requests received from the network. You configure rules for the following types of requests:

- Accounting requests. See "Configuring Incoming Routing Rules for Accounting Requests" for more information.
- Access requests. See "Specifying the Service Broker Component for Dispatching Access Requests" for more information.

Configuring Incoming Routing Rules for Accounting Requests

When an accounting event occurs, a RADIUS client sends an accounting request to Server Broker which acts as a RADIUS server. You can define the IM to which the RADIUS SSU further routes the accounting request based on the local realm set in the request.

You can configure incoming routing rules for accounting requests using the Administration Console or Java MBeans.

Configuring Incoming Routing Rules with the Administration Console

To configure RADIUS SSU Accounting parameters:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Select the **SSU RADIUS** node.
4. In the **SSU RADIUS** tab, click the **Accounting** subtab.
5. At the bottom of the Incoming Routing Rules pane, click the **New** button.

   The New dialog box appears.
6. Fill in the fields of the New dialog box described in Table 6–1.

Table 6–1 RADIUS Accounting Incoming Routing Parameters

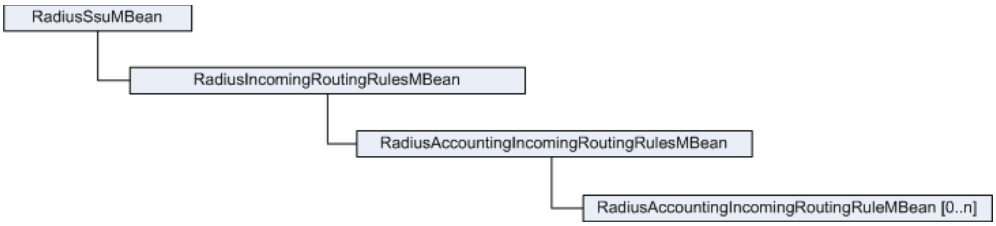
Field	Descriptions
Name	Specifies a unique routing rule name.
Local Realm	Specifies the value to match against the Local Realm. Example: *user-name*`@isp.net` If a RADIUS accounting request arrives containing only a user name but without a Local Realm, the RADIUS SSU discards the request. To prevent the request from being discarded when no Local Realm is specified, set this field to **any**. The RADIUS SSU then forwards the request to the destination specified in the Alias field. Important: When typing **any** into the Local Realm field, you must use only lower case, as follows: **any**. Do not type **Any** or **ANY**.
Alias	Specifies the URL of the destination IM to which the RADIUS message is dispatched. The alias has the following format: `SSU`:*IM-instance-name.IM-type@domain-id* ▪ *IM-instance-name*: IM instance name you specified when you added this IM in the IM configuration pane. ▪ *IM-type*: Type of IM instance. ▪ *domain-id*: Name of the Processing Domain or Processing Domain Group where the relevant IM or application is deployed. This parameter is required only when your Service Broker deployment includes two or more Processing Domains. Use the name given to the domain when it was created. This name is specified by the axia.domain.id property. To set a Processing Domain Group, you must specify the group name. See "Managing Processing Domain Groups" in the *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information. Example: `SSU:imocf.IMOCF@ocsb`.1

7. Click **OK**.

Configuring Incoming Routing Rules with Java MBeans

Figure 6–1 shows the hierarchy of the configuration MBeans that you use to configure incoming routing rules for accounting requests. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–1 Incoming Routing Rules for Accounting Requests Configuration MBeans



Each incoming routing rule is represented by RadiusAccountingIncomingRoutingRuleMBean. You need to create a separate instance of RadiusAccountingIncomingRoutingRuleMBean for each rule.

To create an incoming routing rule for an accounting request:

1. Create an instance of RadiusAccountingIncomingRoutingRuleMBean by invoking the following operation of RadiusAccountingIncomingRoutingRulesMBean:

   ```
   ObjectName createRadiusAccountingIncomingRoutingRule(string IncomingRuleName)
   ```

2. Set the following attributes of the newly created instance of RadiusAccountingIncomingRoutingRuleMBean:

   - LocalRealm
   - Alias

   See Table 6–1 for more information about these attributes.

Specifying the Service Broker Component for Dispatching Access Requests

To request authorization for access to a RADIUS server, a Network Access Server (NAS) sends a RADIUS access request to Service Broker which acts as a RADIUS server. You can specify the Service Broker component to which the RADIUS SSU dispatches the access request.

You can specify this component using the Administration Console or Java MBeans.

Specifying the Service Broker Component with the Administration Console

To specify the Service Broker component:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Select the **SSU RADIUS** node.
4. In the **SSU RADIUS** tab, click the **Access** subtab.
5. In the **Radius Access Inbound Destination** field, enter the address of the Service Broker component to which you want to dispatch the RADIUS Access request.

   The address has the following format: `ssu:`*domain*

   domain: The name of the domain to which the request is despatched.

For example: `ssu:ocsb`

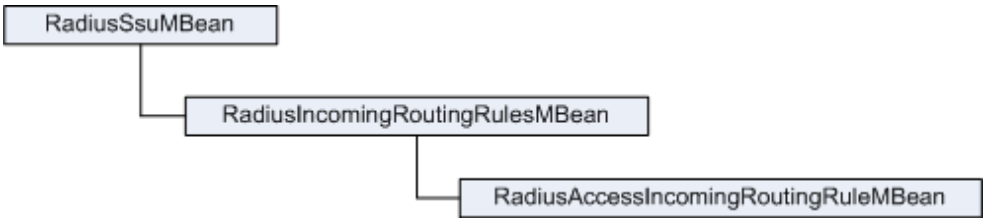
If you leave this field empty, the request is not routed through Service Broker.

6. Click **Apply**.

Specifying the Service Broker Component with Java MBeans

Figure 6–2 shows the hierarchy of configuration MBeans that you use to specify the Service Broker component to which the RADIUS SSU dispatches access requests. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–2 Access Requests Configuration MBeans



To specify the Service Broker component:

- Set the AccessDestination attribute of RadiusAccessIncomingRoutingRuleMBean to the address of the Service Broker component to which the RADIUS SSU needs to route the access request. See

  The address has the following format: `ssu:`*domain*

  domain: The name of the domain to which the request is despatched.

  For example: `ssu:ocsb`

  If you leave this field empty, the request is not routed through Service Broker.

Providing a Custom Dictionary

By default, Service Broker uses the standard RADIUS dictionary defined in the RFC 2865 (see `http://www.ietf.org/rfc/rfc2865.txt` for more information). If you need Service Broker to recognize additional vendor-specific AVPs, you can provide Service Broker with a custom dictionary.

You can provide a custom dictionary file using the Administration Console or Java MBeans.

Providing a Custom Dictionary File with the Administration Console

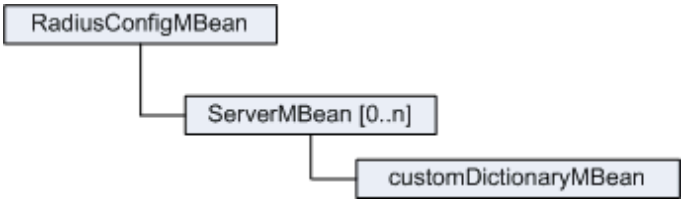
To provide a custom dictionary file:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS Custom Dictionary** tab.
5. In the **Custom dictionary file** field, enter the path of the custom dictionary file located on your local file system.

Providing a Custom Dictionary File with Java MBeans

Figure 6–3 shows the hierarchy of the configuration MBeans that you use to provide a custom dictionary file. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–3 Custom Dictionary Configuration MBeans



To provide a custom dictionary file:

1. Create an instance of customDictionaryMBean by invoking the following operation of ServerMBean:

   ```
   void addCustomDictionary()
   ```

2. Set the **path** attribute of the newly created instance of customDictionaryMBean to the path of your custom dictionary file located on your local file system.

Configuring Server Parameters

To receive RADIUS authentication and accounting requests from the network, you configure the following:

- Server parameters, which define how the RADIUS SSU receives RADIUS requests. See "Configuring Server Parameters" for more information.
- NAS port range, which defines the range of NASs ports from which the RADIUS SSU can receive accounting authentication and accounting requests. See "Specifying the NAS Port Range" for more information.

Configuring Server Parameters

You set up server parameters to define how the RADIUS SSU receives RADIUS requests. You can set up server parameters using the Administration Console or Java MBeans.

Configuring Server Parameters with the Administration Console

To configure server parameters:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab.
5. Click the **Server Configuration** subtab and then the **Server** tab.
6. Click **New**.

   The New dialog box appears.
7. Fill in the fields described in Table 6–2.

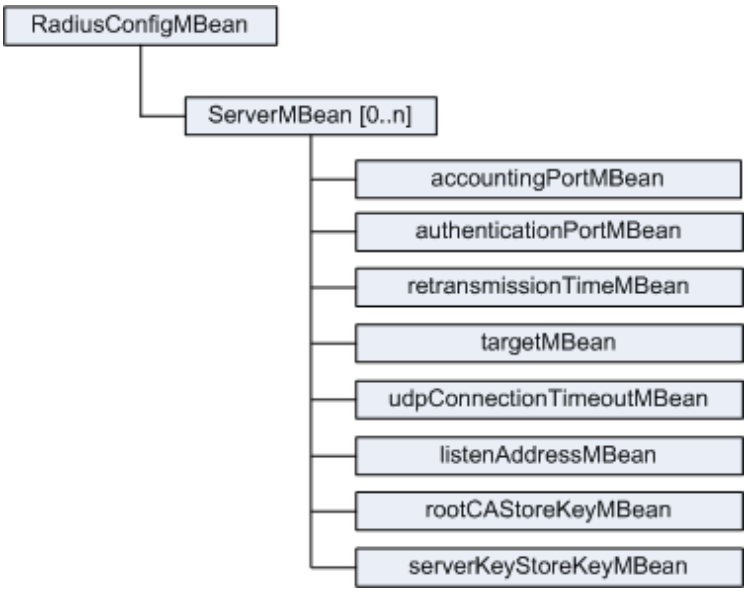
Table 6–2 Server Parameters

Field	Description
Target managed server	Specifies the target managed server.
IP Address	Specifies the IP address that the RADIUS SSU uses to listen for RADIUS messages.
Authentication Port	Specifies the port that the RADIUS SSU uses to receive RADIUS authentication messages.
Accounting Port	Specifies the port that the RADIUS SSU uses to receive RADIUS accounting messages.
UDP Connection timeout	Specifies the UDP connection timeout in seconds.
Retransmission detection time	Specifies the period of time during which the RADIUS SSU considers incoming RADIUS messages retransmissions if these messages have the same ID received and are sent by the same peer. The RADIUS SSU ignores these messages. If you set the retransmissionTime parameter to 0, the RADIUS SSU does not recognize these messages as retransmissions.
Root CA Store key	Specifies the root CA keystore key. You provide this key to the credential store that contains root CA certificates.
Server Key Store key	Specifies the server keystore. You provide this key to the credential store that contains server certificates.

Configuring Server Parameters with Java MBeans

Figure 6–4 shows the hierarchy of the configuration MBeans that you use to configure server parameters. See "RADIUS SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 6–4 RADIUS Server Configuration MBeans



When you create an instance of ServerMBean, the set of child MBeans are created. Each child MBean represents a single parameter of the server.

To set up a server:

1. Create an instance of ServerMBean by invoking the following operation of RadiusConfigMBean:

   ```
   void addServer()
   ```

   ServerMBean is created with the following child MBeans:

   - accountingPortMBean
   - authenticationPortMBean
   - retransmissionTimeMBean
   - targetMBean
   - udpConnectionTimeoutMBean

2. Create additional child MBeans of ServerMBean by invoking the following operations of ServerMBean:

   ```
   void addListenAddress()
   void addRootCAStoreKey()
   void addServerKeyStoreKey()
   ```

3. Set the attributes of the child MBeans as described in Table 6–3.

Table 6–3 ServerMBean Child MBeans and Their Attributes

MBean	Attribute
listenAddressMBean	listenAddress
rootCAStoreKeyMBean	rootCAStoreKey
serverKeyStoreKeyMBean	serverKeyStoreKey
authenticationPortMBean	authenticationPort
accountingPortMBean	accountingPort
udpConnectionTimeoutMBean	udpConnectionTimeout
retransmissionTimeMBean	retransmissionTime
targetMBean	target

See Table 6–2 for more information about these attributes.

Specifying the NAS Port Range

You can configure Service Broker to receive RADIUS requests from those NASs whose port is a certain range. To specify the range, you define the lower and upper limits of the range.

You can specify the NAS port range using the Administration Console or Java MBeans.

Specifying the NAS Port Range with the Administration Console

To specify the port range:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab and then the **Server Configuration** subtab.

5. In the **Valid NAS Port Range** tab, fill in the fields as described in Table 6–4.

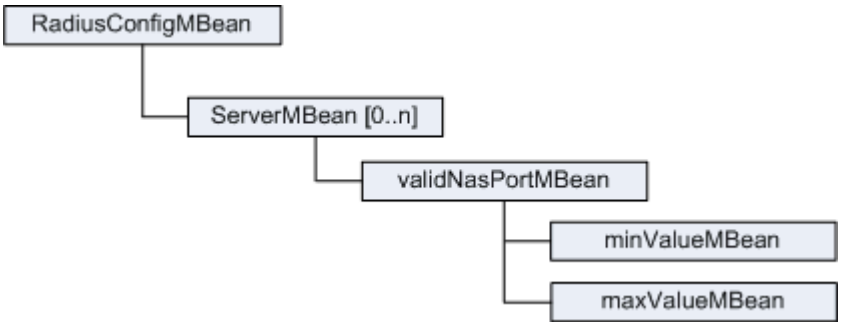
Table 6–4 Valid NAS Port Range Parameters

Field	Description
Min Value of NAS Port	Specifies the lower limit of the range.
Max Value of NAS Port	Specifies the upper limit of the range.

Specifying the NAS Port Range with Java MBeans

Figure 6–6 shows the hierarchy of the configuration MBeans that you use to specify the NAS port range. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–5 NAS Range Configuration MBeans



To specify the NAS port range:

1. Create an instance of validNasPortMBean by invoking the following operation of ServerMBean:

   ```
   void addValidNasPort()
   ```

   validNasPortMBean is created with the following child MBeans:

   - minValueMBean
   - maxValueMBean

2. Set the attributes of the child MBeans as described in Table 6–5.

Table 6–5 validNasPort Child MBeans and Their Attributes

MBean	Attribute
minValueMBean	minValue
maxValueMBean	maxValue

Setting Up RADIUS Clients

A RADIUS client is a a network entity from which the RADIUS SSU receives accounting and access requests. To enable the RADIUS SSU to establish communication with RADIUS clients, you need to define a set of settings for each RADIUS client.

This set of settings consists of the following:

- Client profiles, which contain the address, NAS identifier, and shared secret keys that the RADIUS SSU uses for communication with the RADIUS client. See "Setting Up a Client Profile" for more information.
- AVPs that the RADIUS SSU needs to copy from a request to a response. See "Specifying AVPs to Be Copied from a Request to a Response" for more information.

The Online Mediation Controller is provided with the client profile and AVPs for a default client profile. You can use the default profile as is. In addition, you can set up configuration settings for additional RADIUS clients.

Setting Up a Client Profile

To receive RADIUS requests, you define the RADIUS clients from which the RADIUS SSU receives accounting and access requests.

A client profile defines RADIUS clients from which the RADIUS SSU receives accounting and access requests.

The Online Mediation Controller is provided with a default client profile. You can edit the settings of the default profile and create new profiles using the Administration Console or Java MBeans.

Setting Up a Client Profile with the Administration Console

To set up a client profile:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab and then the **Client Profile** tab.
5. Click the **Client Profile** tab.
6. Click **New**.

   The New dialog box appears.
7. Fill in the fields described in Table 6–6.

Table 6–6 Client Profile Parameters

Field	Description
Client Address	Specifies the IP address of the RADIUS client from which the RADIUS SSU receives requests. If you want to define a range of addresses to receive requests from a group of RADIUS clients, you can use a regular expression. For example, if you want to define that the RADIUS SSU receives requests from the clients whose IP addresses start from 10.148, you can set the clientAddress parameter to 10.148.*.*

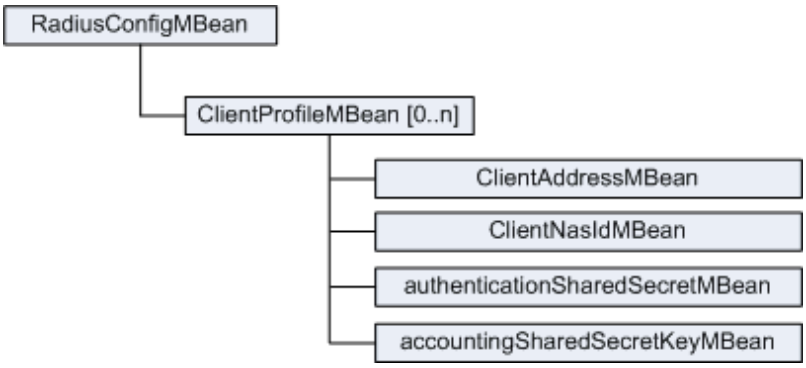
Table 6–6 (Cont.) Client Profile Parameters

Field	Description
Client NAS Identifier	Specified the ID of the Network Access Server (NAS) from which the RADIUS SSU receives accounting and access requests. If you want to define a range of IDs to receive requests from a group of NASs, you can use a regular expression. For example, if you want to define that the RADIUS SSU receives requests from the NASs whose IDs is in the oracle.com domain, you can set the clientNasId to *.oracle.com.
Authentication Shared Secret Key	Specifies the key that you associated with the password that the RADIUS SSU uses for authentication requests. You associate keys and passwords using the Credential Store tab. See "Associating Passwords with Keys" for more information.
Accounting Shared Secret Key	Specifies the key that you associated with the password that the RADIUS SSU uses for accounting requests. You associate keys and passwords using the Credential Store tab. See "Associating Passwords with Keys" for more information.

Setting Up a Client Profile with Java MBeans

Figure 6–6 shows the hierarchy of the configuration MBeans that you use to configure a client profile. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–6 ClientProfile Configuration MBeans



After an instance of ClientProfileMBean is created, you can create the set of child MBeans. Each child MBean represents a single parameter of the profile.

To set up a client profile:

1. Create an instance of ClientProfileMBean by invoking the following operation of RadiusConfigMBean:

   ```
   void addClientProfile()
   ```

2. Create child MBeans of ClientProfileMBean by invoking the following operations of ClientProfileMBean:

   ```
   void addClientAddress()
   void addClientNasId()
   void addCheckAVPFilter()
   void addReturnAVPFilter()
   ```

3. Set the attributes of the child MBeans as described in Table 6–7.

Table 6–7 ClientProfileMBean Child MBeans and Their Attributes

MBean	Attribute
clientAddressMBean	clientAddress
clientNasIdMBean	clientNasId
checkAVPFilterMBean	checkAVPFilter
returnAVPFilterMBean	returnAVPFilter

See Table 6–6 for more information about these attributes.

Specifying AVPs to Be Copied from a Request to a Response

You can specify the AVPS that the RADIUS SSU needs to copy from a request to a response. In the default client profile, the RADIUS SSU copies the following AVPs:

- User-Name
- Acct-Session-Id

You can add more AVPs to the default client profile or to a new client profile that you created using the Administration Console or Java MBeans.

Specifying AVPs to Be Copied with the Administration Console

To specify the AVPs:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab and then the **Client Profile** tab.
5. Click the **Avps to copy from Response to Request** tab.
6. Click **New**.

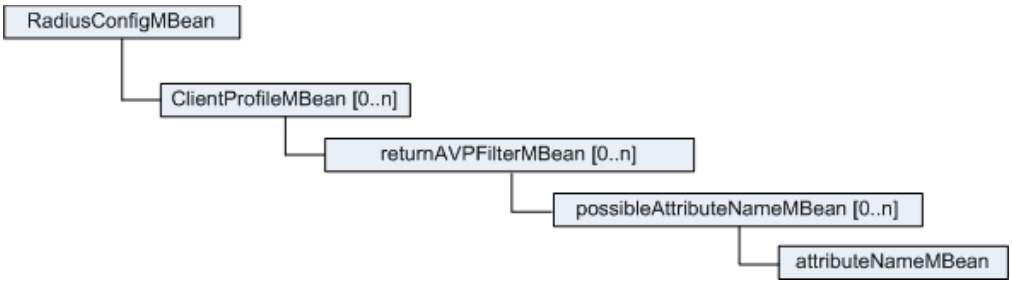
   The New dialog box appears.
7. In the **New** dialog box, in the **Attribute Name** field, enter the name of the AVP that the RADIUS SSU needs to copy.
8. Click **OK**.

   The new AVP appears in the configuration screen.

Specifying AVPs to Be Copied with Java MBeans

Figure 6–7 shows the hierarchy of the configuration MBeans that you use to specify AVPs that the RADIUS SSU needs to copy from a request to a response. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–7 RADIUS AVPs Configuration MBeans



To specify the AVPs that the RADIUS SSU needs to copy:

1. Create an instance of returnAVPFilterMBean by invoking the following operation of ClientProfileMBean:

   ```
   void addReturnAVPFilter()
   ```

2. Create an instance of possibleAttributeNameMBean by invoking the following operation of returnAVPFilterMBean:

   ```
   void addPossibleAttributeName()
   ```

   The new instance of possibleAttributeNameMBean is created with the child MBean attributeNameMBean.

3. Set the **attributeName** of the attributeNameMBean to the name of the AVP that the RADIUS SSU needs to copy from the request to a response.

Configuring Proxy Realm

You can configure the RADIUS SSU to forward accounting and access requests to a proxy server. You can define the proxy server to which the RADIUS SSU routes a request based on the user name specified in the incoming request.

When you configure a proxy server, you define the following:

- Proxy realm, which defines a realm to which the RADIUS SSU routes a request based on the User-Name AVP. See "Configuring Proxy Realm" for more information.
- Target server, which defines the server address and ports in the specified realm. See "Configuring Target Servers" for more information.

Configuring Proxy Realm

A proxy realm defines the realm to which the RADIUS SSU routes an accounting or access request. You can set up conditions for routing a request to a specific realm based on the User-Name AVP specified in the incoming request.

You configure a proxy realm using the Administration Console or Java MBeans.

Configuring Proxy Realm with the Administration Console

To configure a proxy realm:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.

4. Click the **RADIUS** tab.
5. Click the **Proxy Realm** subtab and then the **Proxy Realm** tab.
6. Click **New**.

   The New dialog box appears.
7. Fill in the fields described in Table 6–8.

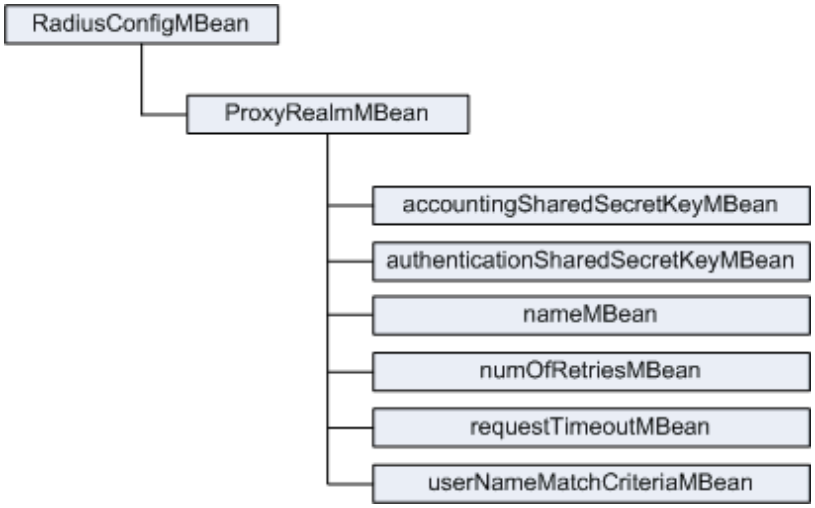
Table 6–8 Proxy Realm Parameters

Field	Description
Name	Specifies the name of the proxy realm.
Username Match Criteria	Specifies the User-Name AVP to be set in the incoming request. If this AVP matches the value of the userNameMatchCriteria parameter, the RADIUS SSU routes the request to the realm specified in the name parameter. To define a range of possible names, you can use regular expressions.
Authentication Shared Secret Key	Specifies the key that you associated with the password that the RADIUS SSU uses for authentication requests. You associate keys and passwords using the Credential Store tab. See "Associating Passwords with Keys" for more information.
Accounting Shared Secret Key	Specifies the key that you associated with the password that the RADIUS SSU uses for accounting requests. You associate keys and passwords using the Credential Store tab. See "Associating Passwords with Keys" for more information.
Request Timeout	Specifies the period of time, in seconds, that the RADIUS SSU waits for a response from the target RADIUS server.
Number Of Retries	Specifies the number of attempts that the RADIUS SSU tries to send a RADIUS request to the target RADIUS server.

Configuring Proxy Realm with Java MBeans

Figure 6–8 shows the hierarchy of the configuration MBeans that you use to configure a proxy realm. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–8 Proxy Realm Configuration MBeans



When you create an instance of ProxyRealmMBean, the set of child MBeans is automatically created. Each child MBean represents a single parameter of the proxy realm.

To set up a proxy realm:

1. Create an instance of ProxyRealmMBean by invoking the following operation of RadiusConfigMBean:

   ```
   void addProxyRealm()
   ```

   ProxyRealmMBean is created with the following child MBeans:

   - accountingSharedSecretKeyMBean
   - authenticationSharedSecretKeyMBean
   - nameMBean
   - numOfRetriesMBean
   - requestTimeoutMBean
   - userNameMatchCriteriaMBean

2. Set the attributes of the child MBeans as described in Table 6–9.

Table 6–9 ProxyRealmMBean Child MBeans and Their Attributes

MBean	Attribute
accountingSharedSecretKeyMBean	accountingSharedSecretKey
authenticationSharedSecretKeyMBean	authenticationSharedSecretKey
nameMBean	name
numOfRetriesMBean	numOfRetries
requestTimeoutMBean	requestTimeout
userNameMatchCriteriaMBean	userNameMatchCriteria

See Table 6–8 for more information about these attributes.

Configuring Target Servers

A target server defines a server in the realm to which the RADIUS SSU routes an accounting or access request when the User-Name AVP set in the request matches the value set in the proxy realm configuration.

You can configure a target server using the Administration Console or Java MBeans.

Configuring Target Servers with the Administration Console

To configure a target server:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab.
5. Click the **Proxy Realm Configuration** subtab and then the **Target Servers** tab.

6. In the **Parent** list, select the proxy realm for which you set up the server. The list displays the proxy realms that you configured using the ProxyRealm tab. See "Configuring Proxy Realm with the Administration Console" for more information.
7. Click **New**.

   The New dialog box appears.
8. Fill in the fields described in Table 6–10.

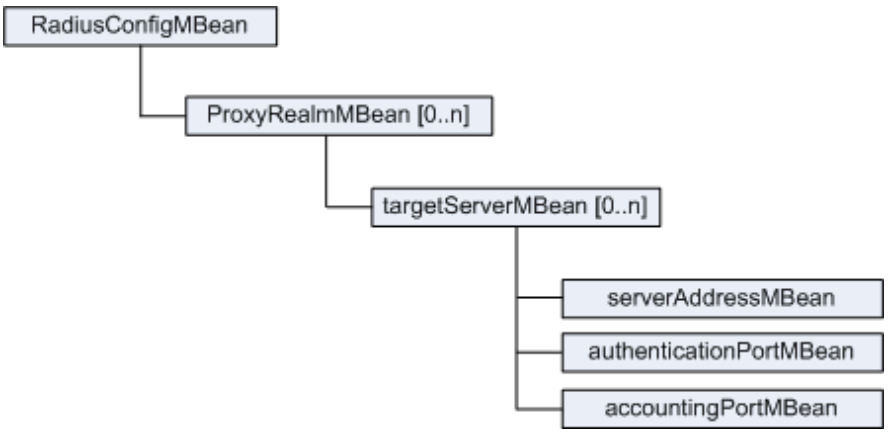
Table 6–10 TargetServers Parameters

Field	Description
Server Address	Specifies the IP address of the proxy server.
Authentication port	Specifies the port that the RADIUS SSU uses to receive RADIUS authentication messages.
Accounting port	Specifies the port that the RADIUS SSU uses to receive RADIUS access messages.

Configuring Target Servers with Java MBeans

Figure 6–9 shows the hierarchy of the configuration MBeans that you use to configure target servers. See "RADIUS SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 6–9 TargetServer Configuration MBeans



When you create an instance of targetServerMBean, the set of child MBeans is automatically created. Each child MBean represents a single parameter of the target server.

To set up a target server:

1. Create an instance of ProxyRealmMBean by invoking the following operation of RadiusConfigMBean:

   ```
   void addProxyRealm()
   ```

2. Create an instance of targetServerMBean by invoking the following operation of ProxyRealmMBean:

   ```
   void addTargetServer()
   ```

   targetServerMBean is created with the following child MBeans:

- serverAddressMBean
- authenticationPortMBean
- accountingPortMBean

3. Set the attributes of the child MBeans as described in Table 6–11.

Table 6–11 targetServerMBean Child MBeans and Their Attributes

MBean	Attribute
serverAddressMBean	serverAddress
authenticationPortMBean	authenticationPort
accountingPortMBean	accountingPort

See Table 6–10 for more information about these attributes.

Configuring the Credential Store

To allow Service Broker to establish connection with a RADIUS client or server, you need to provide a valid password.Service Broker stores these passwords in the credential store.

Associating Passwords with Keys

Each password that Service Broker stores in the credential store must have a key. You define a key when specifying a password.Then you use the key when setting up a RADIUS client or a server in the RADIUS SSU. You can use the same key for multiple clients or servers. For example, this can be useful when you use the same password for authentication, accounting, and proxy configuration.

To associate a password with a key:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab.
5. Click the **Credential Store** subtab.
6. In the **Password** area, in the **Key** field, enter a key that you want to associate with the password.
7. In the **Password** field, enter the password for connection with a RADIUS client or server.
8. Do one of the following:
   - If you want the RADIUS SSU to include credentials in outgoing requests and thus, authenticate to an external entity, clear the **One-way** check box.
   - If you want the RADIUS SSU to include credentials in incoming requests only, select the **One-way** check box.

Setting Up the Key Store Password and URL

The RADIUS SSU stores connections and their passwords in the encrypted file known as Credential Store. You specify the name and location of the Credential Store file

using the Java system property **javax.net.ssl.keyStore**. See "Configuring Security", *Oracle Communications Server Broker System Administrator's Guide* for more information.

You can set up the key store password using the Administration Console or Java MBeans.

Setting Up the Key Store Password with the Administration Console

To set up the key store password:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab and then the **Credential Store** subtab.
5. In the **KeyStore** area, fill in the fields described in Table 6–12:

Table 6–12 KeyStore Fields

Field	Descriptions
Key	Specifies a key under which the keystore is stored.
KeyStore Password	Specifies the password that protects the keystore.
KeyStore URL	Specifies the URL of the keystore file on a hard drive.

Setting Up the KeyStore Password with Java MBeans

To set up the credential store password with Java MBeans, you use the **CredentialStoreMBean**.

To set up the key store password:

- Invoke the following operation of **CredentialStoreMBean**:

  ```
  void setKeystore (string Key, string KeystorePassword, string KeystoreURL)
  ```

  See Table 6–12 for more information about these parameters.

Managing Keys in the Credential Store

You can check whether a key exists in the Credential store and delete keys. You can manage keys using the Administration Console or Java MBeans.

Managing Keys in the Credential Store with the Administration Console

To check whether a key exists in the credential store:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **RADIUS SSU** node.
4. Click the **RADIUS** tab and then the **Credential Store** subtab.
5. In the General area, in the Key field, enter the key whose existence you want to check.
6. Click **Contains Key?**.

   The message which informs you whether the key exists, appears.

7. To close the message, click **OK**.

To delete a specified key from the credential store:

- In the **Credential Store** tab, in the General area, in the Key field, enter the key and then click **Delete Key**

To delete all keys from the credential store:

- In the **Credential Store** tab, in the General area, click **Delete All Keys.**

Managing Keys in the Credential Store with Java MBeans

To manage keys in the credential store with Java MBeans, you use the **CredentialStoreMBean**.

To check whether a key exists in the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
boolean containsKey (string Key)
```

To delete a specified key from the credential store:

- Invoke the following operations of **CredentialStoreMBean**

```
void deleteKey (string Key)
```

To delete all keys from the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
void clear ()
```

RADIUS SSU Configuration MBeans Reference

The following sections provide reference information for the RADIUS SSU configuration MBeans.

> **Note:** MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

RadiusSsuMBean

RadiusSsuMBean is a container for instances of RadiusIncomingRoutingRulesMBean.

Object Name

com.convergin:Type=RadiusSsu,Version=*MBean_Version*,Location=AdminServer,Name=ssuradius.ssuradius

Factory Method

Created automatically

Attributes

None

Operations

ObjectName getRadiusIncomingRoutingRulesMBean()

Returns a reference to the instance of RadiusIncomingRoutingRulesMBean

RadiusIncomingRoutingRulesMBean

RadiusIncomingRoutingRulesMBean represents the container for all the incoming routing rules that define how the RADIUS SSU routes incoming RADIUS requests.

Object Name

com.convergin:Type=RadiusIncomingRoutingRules,Version=*MBean_Version*,Location=AdminServer,Name=ssuradius.radius_inbound_routing

Factory Method

Created automatically

Attributes

None

Operations

ObjectName getRadiusAccountingIncomingRoutingRules()

Returns a reference to the instance of RadiusAccountingIncomingRoutingRulesMBean

ObjectName getRadiusAccessIncomingRoutingRules

Returns a reference to the instance of RadiusAccessIncomingRoutingRulesMBean

RadiusAccountingIncomingRoutingRulesMBean

RadiusAccountingIncomingRoutingRulesMBean represents the container for all the accounting incoming routing rules that define how the RADIUS SSU routes accounting incoming RADIUS messages to internal Server Broker IMs.

Object Name

com.convergin:Type=RadiusAccountingIncomingRoutingRules,Version=MBean_Version,Location=AdminServer,Name=ssuradius.radius_accounting_inbound_routing

Factory Method

Created automatically

Attributes

None

Operations

ObjectName get RadiusAccountingIncomingRoutingRule()

Returns a reference to the instance of RadiusAccountingRoutingRuleMBean

ObjectName createRadiusAccountingIncomingRoutingRule()

Creates an instance of RadiusAccountingIncomingRoutingRuleMBean that enables you to create an individual rule.

void destroyRadiusAccountingIncomingRoutingRule()

Destroys an instance of RadiusAccountingIncomingRoutingRuleMBean

ObjectName lookupRadiusAccountingIncomingRoutingRule()

Searches for an instance of RadiusAccountingIncomingRoutingRuleMBean.

RadiusAccountingIncomingRoutingRuleMBean

Each instance of RadiusAccountingIncomingRoutingRuleMBean represents a single accounting incoming routing rule.

Object Name

com.convergin:Type=RadiusAccountingIncomingRoutingRule,Version=MBean_Version,Location=AdminServer,Name=ssuradius.RuleName

Factory Method

RadiusAccountingIncomingRoutingRules.createRadiusAccountingIncomingRoutingRule()

Attributes

- Name
- LocalRealm
- Alias

For more information on each of these parameters, see Table 6–1.

Operations

None

RadiusAccessIncomingRoutingRuleMBean

Each instance of RadiusAccessIncomingRoutingRuleMBean represents a single access incoming routing rule.

Object Name

com.convergin:Type=RadiusAccessIncomingRoutingRule,Version=MBean_Version,Location=AdminServer,Name=ssuradius.radius_access_inbound_routing

Factory Method

RadiusAccessIncomingRoutingRules.createRadiusAccessIncomingRoutingRule()

Attributes

AccessDestination

Operations

None

RadiusConfigMBean

RadiusConfigMBean is the root MBean for configuring the RADIUS settings.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig

Factory Method

Created automatically.

Attributes

None

Operations

void addServer()

Creates a new instance of ServerMBean.

void removeServer()

Deletes an instance of ServerMBean.

void addClientProfile()

Creates a new instance of ClientProfileMBean.

void removeClientProfile()

Deletes an instance of ClientProfileMBean.

void addServiceMapping()

Creates a new instance of ServiceMappingMBean.

void removeServiceMapping()

Deletes an instance of ServiceMappingMBean.

void addLocalRealm()

Creates a new instance of LocalRealmMBean.

void removeLocalRealm()

Deletes an instance of LocalRealmMBean.

void addProxyRealm()

Creates a new instance of ProxyRealmMBean.

void removeProxyRealm()

Deletes an instance of ProxyRealmMBean.

ServerMBean

ServerMBean is the root MBean that you use to define how the RADIUS SSU receive authentication and accounting requests.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0]

Factory Method

RadiusConfig.addServer()

Attributes

None

Operations

void cloneIt()

Creates a copy of the instance of ServerMBean.

void addListenAddress()

Creates a new instance of ListenAddressMBean.

void removeListenAddress()

Deletes an instance of ListenAddressMBean.

void addRootCAStoreKey()

Creates an instance of RootCAStoreKeyMBean.

void removeRootCAStoreKey()

Deletes an instance of RootCAStoreKeyMBean.

void addServerKeyStoreKey()

Creates a new instance of ServerKeyStoreKeyMBean.

void removeServerKeyStoreKey()

Deletes an instance of ServerKeyStoreKeyMBean.

void addCustomDictionary()

Creates a new instance of CustomDictionaryMBean.

void removeCustomDictionary()

Deletes an instance of CustomDictionaryMBean.

customDictionaryMBean

Using customDictionaryMBean, you specify a custom dictionary file that defines vendor-specific AVPs.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0], name2=customDictionary

Factory Method

server.addCustomDictionary()

Attributes

path

Operations

None

AccountingPortMBean

Specifies the port to receive RADIUS accounting messages.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0],
name2=accountingPort

Factory Method

Created automatically.

Attributes

accountingPort

Operations

None

authenticationPortMBean

Specifies the port to receive RADIUS authentication messages.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0], name2=authenticationPort

Factory Method

Created automatically.

Attributes

authenticationPort

Operations

None

retransmissionTimeMBean

Specifies the time within which the incoming radius messages with the same Id, from the same peer are considered retransmissions and ignored. A value of 0 means, retransmissions are not detected.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0], name2=retransmissionTime

Factory Method

Created automatically.

Attributes

retransmissionTime

Operations

None

targetMBean

Specifies the target managed server.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0],name2=target

Factory Method

Created automatically.

Attributes

target

Operations

None

udpConnectionTimeoutMBean

Specifies the UDP connection timeout value in seconds.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0],
name2=udpConnectionTimeout

Factory Method

Created automatically.

Attributes

udpConnectionTimeout

Operations

None

listenAddressMBean

The IP address to listen for RADIUS messages.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0],
name2=listenAddress

Factory Method

Server.addListenAddress()

Attributes

listenAddress

Operations

None

RootCAStoreKeyMBean

The root CA keystore key. Credential store should be provisioned with the root CA certs with this key

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0], name2=rootCAStoreKey

Factory Method

Server.addRootCAStoreKey()

Attributes

rootCAStoreKey

Operations

None

serverKeyStoreKeyMBean

The server keystore. Credential store should be provisioned with server certs with this key

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=server[0], name2=serverKeyStoreKey

Factory Method

Server.addServerKeyStoreKey()

Attributes

serverKeyStoreKey

Operations

None

ClientProfileMBean

ClientProfileMBean is the root MBean for configuring RADIUS client profiles.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0]

Factory Method

radiusConfig.addClientProfile()

Attributes

None

Operations

void cloneIt()

Creates a copy of the instance of ClientProfileMBean.

void addClientAddress()

Creates a new instance of ClientAddressMBean.

void removeClientAddress()

Deletes an instance of ClientAddressMBean.

void addClientNasId()

Creates an instance of ClientNasIdMBean.

void removeClientNasId()

Deletes an instance of ClientNasIdMBean.

void addCheckAVPFilter()

Creates an instance of CheckAVPFilterMBean.

void removeCheckAVPFilter()

Deletes an instance of CheckAVPFilterMBean.

void addReturnAVPFilter()

Creates an instance of ReturnAVPFilterMBean.

void removeReturnAVPFilter()

Deletes an instance of ReturnAVPFilterMBean.

accountingSharedSecretKeyMBean

Credential store key for the shared secret for accounting. Credential store must be configured with this key and the password. Password will be retrieved from the Credential store at runtime using this key.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0], name2=accountingSharedSecretKey

Factory Method

Created automatically.

Attributes

accountingSharedSecretKey

Operations

None

authenticationSharedSecretKeyMBean

Credential store key for the shared secret for authentication. Credential store must be configured with this key and the password. Password will be retrieved from the Credential store at runtime using this key.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0], name2=authenticationSharedSecretKey

Factory Method

Created automatically.

Attributes

authenticationSharedSecretKey

Operations

None

clientAddressMBean

A regular expression that matches the IP addresses of a group of RADIUS clients.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0],
name2=clientAddress

Factory Method

ClientProfile.addClientAddress()

Attributes

clientAddress

Operations

None

clientNasIdMBean

A regular expression that matches the NAS-Identifier field of a group of RADIUS clients.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0],name2=clientNasId

Factory Method

ClientProfile.addClientNasId()

Attributes

clientNasId

Operations

None

returnAVPFilterMBean

returnAVPFilterMBean is a root MBean that you use to specify the AVPs that the RADIUS SSU needs to copy from a request to a response. For each AVP, you need to create a separate instance of possibleAttributeNameMBean.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=*MBean_Version*,name0=radiusConfig,name1=clientProfile[0], name2=returnAVPFilter[0]

Factory Method

clientProfile.addReturnAVPFilter()

Attributes

None

Operations

void cloneIt()

Creates a copy of the current instance of returnAVPFilterMBean.

void addPossibleAttributeName()

Creates an instance of possibleAttributeNameMBean.

void removePossibleAttributeName(int AttributeNameID)

Destroys the specified instance of possibleAttributeNameMBean.

possibleAttributeNameMBean

possibleAttributeNameMBean represents a single AVP that the RADIUS SSU needs to copy from a request to a response. For each AVP, you need to create a separate instance of possibleAttributeNameMBean using returnAVPFilterMBean. possibleAttributeNameMBean is created with an instance of attributeNameMBean that you use to specify the name of an AVP.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0], name2=returnAVPFilter[0],name3=possibleAttributeName[0]

Factory Method

returnAVPFilter.addPossibleAttributeName()

Attributes

None

Operations

void cloneIt()

Creates a copy of the current instance of possibleAttributeNameMBean.

attributeNameMBean

Using attributeNameMBean, you specify the name of an AVP that the RADIUS SSU needs to copy from a request to a response.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=clientProfile[0], name2=returnAVPFilter[0],name3=possibleAttributeName[0],name4=attributeName

Factory Method

Created automatically.

Attributes

attributeName

See "Specifying AVPs to Be Copied from a Request to a Response" for more information about this attribute.

Operations

None

proxyRealmMBean

proxyRealmMBean is the root MBean for configuring proxy realm.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0]

Factory Method

radiusConfig.addProxyRealm()

Attributes

None

Operations

void cloneIt()

Creates a copy of the instance of proxyRealmMBean.

void addTargetServer()

Creates an instance of TargetServerMBean.

void removeTargetServer()

Deletes an instance of TargetServerMBean.

accountingSharedSecretKeyMBean

Credential store key for the shared secret for accounting. Credential store must be configured with this key and the password. Password will be retrieved from the Credential store at runtime using this key.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0], name2=accountingSharedSecretKey

Factory Method

Created automatically.

Attributes

accountingSharedSecretKey

Operations

None

authenticationSharedSecretKeyMBean

Credential store key for the shared secret for authentication. Credential store must be configured with this key and the password. Password will be retrieved from the Credential store at runtime using this key.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0], name2=authenticationSharedSecretKey

Factory Method

Created automatically.

Attributes

authenticationSharedSecretKey

Operations

None

nameMBean

The name of the proxy realm.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0],
name2=name

Factory Method

Created automatically.

Attributes

name

Operations

None

numOfRetriesMBean

Number of attempts to send a RADIUS request to the target RADIUS server.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0],
name2=numOfRetries

Factory Method

Created automatically.

Attributes

numOfRetries

Operations

None

requestTimeoutMBean

Timeout value in seconds to wait for response from the target RADIUS server.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0],name2=requestTimeout

Factory Method

Created automatically.

Attributes

requestTimeout

Operations

None

userNameMatchCriteriaMBean

A regular expression for mapping RADIUS requests to the realm based on the User-Name attribute.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0], name2=userNameMatchCriteria

Factory Method

Created automatically.

Attributes

userNameMatchCriteria

Operations

None

targetServerMBean

targerServerMBean enables you to define parameters of the server in the realm to which the RADIUS SSU routes an accounting or access request when the User-Name AVP set in the request matches the value set in the proxy realm configuration.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0], name2=targetServer[0]

Factory Method

Created automatically.

Attributes

None

Operations

void cloneIt()

Creates a copy of the instance of proxyRealmMBean.

serverAddressMBean

serverAddressMBean enables you to specify the IP address of the proxy server.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0],name2=targetServer[0],name3=serverAddress

Factory Method

Created automatically.

Attributes

serverAddress

Operations

None

authenticationPortMBean

authenticationPortMBean enables you to specify the port that the RADIUS SSU uses to receive RADIUS authentication messages.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0],name2=targetServer[0],name3=authenticationPort

Factory Method

Created automatically.

Attributes

authenticationPort

Operations

None

accountingPortMBean

accountingPortMBean enables you to specify the port that the RADIUS SSU uses to receive RADIUS access messages.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=MBean_Version,name0=radiusConfig,name1=proxyRealm[0], name2=targetServer[0],name3=accountingPort

Factory Method

Created automatically.

Attributes

accountingPort

Operations

None

CredentialStoreMBean

CredentialStoreMBean enables you to set up passwords for connections with RADIUS clients and the credential store. In addition, you can use CredentialStoreMBean to manage keys in the credential store.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.radius.adapter,version=*MBean_Version*,name0=CredentialStore

Factory Method

Created automatically

Attributes

None

Operations

void setPassword()

Sets a password for the specified key

boolean validatePassword()

Validate a password against a password stored for the specified key

void setKeystore()

Validate a password against a password stored for the specified key

boolean containsKey()

Return true if the key exists in the credential store

void deleteKey()

Delete a key

void clear()

Remove all entries

7

Configuring PCP Signaling Server Units

This chapter describes how to configure the Service Broker Portal Communications Protocol (PCP) Signaling Server Unit (SSU) using the Service Broker Administration Console and Java MBeans.

About the PCP SSU

Service Broker acts as an Oracle Communications BRM client application and uses the PCP SSU to communicate with it. The communication is done through a proprietary Oracle Communications BRM protocol, the PCP.

The PCP SSU receives charging requests from internal Service Broker components, such as IM-OFCF PCP and IM-OCF PCP, and routes these request to BRM applications through PCP.

To access Oracle Communications BRM, the PCP SSU must use a BRM client application account. The PCP SSU uses the default root account created when BRM is installed, and its password. See "Configuring Login Names and Passwords for BRM Access" in the chapter "Implementing System Security" in *Oracle Communications Billing and Revenue Management System Administration Guide*.

The PCP SSU uses connection pools to communicate with BRM. A connection pool is a set of connections maintained between the PCP SSU and the BRM Connection Manager (CM). Each Oracle Communications BRM CM is running on a different physical address and listens to PCP request on a different port. See "About Connection Pooling" in *Oracle Communications Billing and Revenue Management System Administration Guide*. On the BRM side, an incoming request is assigned a connection from the connection pool and uses the connection to perform operations. When the operation completes, the connection is returned to the pool.

When configuring the PCP SSU connectivity with BRM, you define a number of connection pools, each communicates with one CM. Oracle recommends that you define two or more connection pools, for redundancy and high availability.

Connection pools need to be secured with passwords. You set a password in both the PCP SSU and Oracle Communications BRM. You have to configure the same password for a connection pool that you define in the PCP SSU, and the related CM that you configure on Oracle Communications BRM.

Each connection pool is considered a PCP network entity. If one or more PCP network entities act as one logical Oracle Communications BRM application, you assign one alias to those network entities. Service Broker IMs sending PCP requests to an Oracle Communications BRM application, use the alias to specify the destination BRM application. The PCP SSU routing the requests to their destination, provides a measure

of redundancy, distributing the PCP requests among the different PCP network entities belonging to the same destination BRM application.

The PCP SSU implements a heartbeat mechanism, regularly sending requests to PCP network entities to check their availability. Not receiving a response within a configured time interval denotes that the PCP network entity is inactive. The PCP SSU stops sending requests to inactive network entities, but continues checking their availability every few seconds. You configure the heartbeat mechanism for each PCP network entity separately, when you define the network entity.

After configuring connection pools and destination Oracle Communications BRM applications, you can also optionally change the default configuration of PCP transactions.

Configuring PCP Transactions

You use either the Administration Console or Java MBeans to configure PCP transactions.

Configuring PCP Transactions with the Administration Console

To configure PCP Transaction:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Select the **SSU PCP** node.
4. In the **PCP** tab, select the **Opcode** tab.
5. Enter the fields described in Table 7–1.

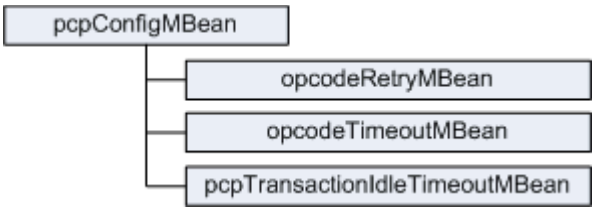
Table 7–1 Opcodes Parameters

Field	Descriptions
opcodeRetry	Specifies the number of attempts that the PCP SSU tries to execute a BRM opcode. Default value: 3
opcodeTimeout	Specifies the opcode timeout by default Default value: 5000
pcpTransactionIdleTimeout	Specifies the PCP transaction timeout. Default value: 10000

Configuring PCP Transactions with Java MBeans

Figure 7–1 shows the hierarchy of the configuration MBeans that you use to configure PCP transactions. See "PCP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 7–1 Opcodes Configuration MBeans



To configure opcodes:

1. Create an instance of **OpcodeRetryMBean** by invoke the following operation of **pcpConfigMBean**:

   ```
   void addOpcodeRetry()
   ```

   The new instance of **opcodeRetryMBean** is created.

2. Set the **opcodeRetry** attribute of **opcodeRetryMBean** as described in Table 7–1.

3. Set the **opcodeTimeout** attribute of **opcodeTimeoutMBean** as described in Table 7–1.

4. Create a new instance of **PcpTransactionIdleTimeoutMBean** by invoking the following operation of **pcpConfigMBean**:

   ```
   void addPcpTransactionIdleTimeout()
   ```

   The new instance of **pcpTransactionTimeoutMBean** is created.

5. Set the **pcpTransactionTimeout** attribute of **pcpTransactionTimeoutMBean** as described in Table 7–1.

Setting Up Connection Pools

To set up a connection pool:

1. Define the connection pool. See "Defining Connection Pools".

2. Secure the connection pool. See "Securing Connection Pools". After securing the connection pool, you can further manage the connection pool credentials. See "Managing Connection Pool Credentials".

Defining Connection Pools

To define a connection pool you need to specify the physical address and port of the Oracle Communications BRM CM providing connections, and a assign an identifier, that is a Pool ID, to the connection.

You configure connection pools using either the Administration Console or Java MBeans.

Configuring Connection Pools with the Administration Console

To configure connection pools:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.

2. Expand the **Signaling Tier** node.

3. Select the **SSU PCP** node.

4. In the **PCP** tab, select the **Connection Pools** tab.

5. Click the **New** button. The New dialog box appears.
6. Enter the fields described in Table 7–2.

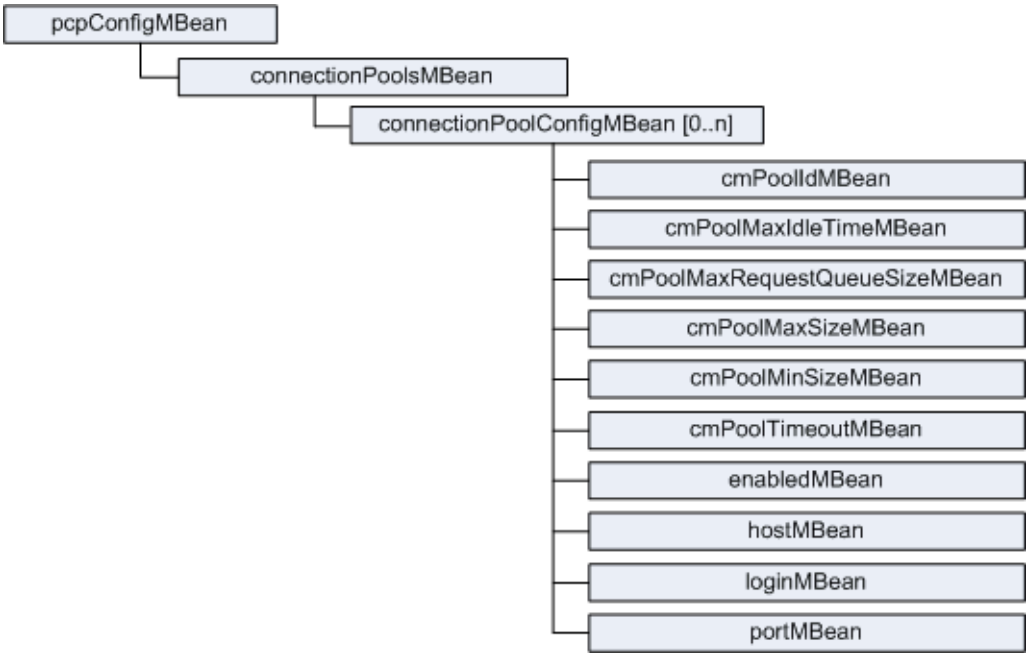
Table 7–2 Connection Pool Parameters

Field	Descriptions
Pool ID	A unique connection pool identifier.
BRM CM Host	The name or IP address of the machine running the Oracle Communications BRM CM. See "Using Configuration Files to Connect and Configure Components" in *Oracle Communications Billing and Revenue Management System Administrator's Guide*.
BRM CM Port	The port number of the Oracle Communications BRM CM on the host machine. See "Using Configuration Files to Connect and Configure Components" in *Oracle Communications Billing and Revenue Management System Administrator's Guide*.
BRM CM Login ID	*account-name.database-number* Where: *account-name* is the name of the BRM client application account that Service Broker uses to access Oracle Communications BRM. See "Configuring Login Names and Passwords for BRM Access" in the chapter "Implementing System Security" in *Oracle Communications Billing and Revenue Management System Administrator's Guide*. *database-number* is the database number that you configured when you installed the BRM application. Default value: root.0.0.0.1 See "Installing BRM" in Oracle Communications Billing and Revenue Management Installation Guide.
Maximum Connections	Specifies the maximum number of connections in the pool. Default value: 8
Minimum Connections	Specifies the minimum number of connections in the pool. Default value: 4
Request Timeout	Specifies the time that the PCP SSU waits for establishing a connection.
Maximum Idle Time	Specifies the maximum idle time for a connection. When the specified time expires, the PSP SSU closes the connection.
Request Queue Size	Specifies the maximum size of the queue. When the maximum size is reached, the PCP SSU drops all further requests.
Enabled	Specifies whether the connection is enabled. You can use this field to disable a connection pool, for example, for maintenance. Possible values: ■ true ■ false Default value: true

Configuring Connection Pools with Java MBeans

Figure 7–2 shows the configuration MBeans that you use to configure BRM connections. See "PCP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 7–2 Connection Pools Configuration MBeans



Each connection is represented by **connectionPoolConfigMBean**. You need to create a separate instance of **connectionPoolConfigMBean** for each connection. After you created an instance of **connectionPoolConfigMBean**, you can create its child MBeans. Each of these child MBeans represents a specific parameter of the connection. For example, **cmPoolIdMBean** has the attribute cmPoolId which represents the Pool ID parameter.

All connections are grouped and represented by **connectionPoolsMBean**.

To create a connection:

1. Create an instance of connectionPoolsMBean by invoking the following operation of pcpConfigMBean:

   ```
   void addConnectionPools()
   ```

   The new instance of connectionPoolsMBean is created.

2. Create a instance of connectionPoolConfigMBean by invoking the following operation of connectionPoolsMBean:

   ```
   void addConnectionPoolConfig()
   ```

   The new instance of connectionPoolMBean with a set of child MBeans is created.

3. Set the attributes of the child MBeans as described in Table 7–3.

Table 7–3 BRM Connection Child MBeans Attributes

MBean	Attribute
cmPoolIdMBean	cmPoolId
hostMBean	host
portMBean	port
loginMBean	login
cmPoolMaxSizeMBean	cmPoolMaxSize
cmPoolMinSizeMBean	cmPoolMinSize
cmPoolTimeoutMBean	cmPoolTimeout
cmPoolMaxIdleTimeMBean	cmPoolMaxIdleTime
cmPoolMaxRequestQueueSizeMBean	cmPoolMaxRequestQueueSize
enabledMBean	enabled

Securing Connection Pools

To access the BRM using a connection pool, the PCP SSU is using a BRM client application account. You configure the BRM client application name when you set up a connection pool. You use the PCP SSU Credential Store to configure and save the BRM client application password.

You must specify a password for each connection pool. The password you configure is the password of the BRM client application account used by the connection pool to access the BRM.

You can set up connection pool passwords using the Administration Console or Java MBeans.

Setting Up a Connection Pool Password with the Administration Console

To set up a connection pool password:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Select the **SSU PCP** node.
4. In the **PCP** tab, select the **Credential Store** tab.
5. In the Password area, enter in the fields described in Table 7–4:

Table 7–4 Connection Pool Password

Field	Descriptions
Key	A connection pool identifier. The identifier that you assigned to the connection pool, in the field Pool ID, when you initially defined the connection pool.
Password	The password of the BRM client application account used by the connection pool to access the BRM. This is the password of the account that you configured in the BRM CM Login ID, when you initially defined the connection pool.

Table 7–4 (Cont.) Connection Pool Password

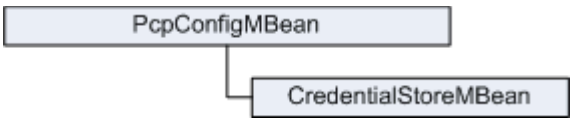
Field	Descriptions
One-way	Always uncheck this box. The PCP SSU should be able retrieve the connection pool password from the Credential Store and use it when sending requests to Oracle Communications BRM.

6. Click the **Set Password** button.

Setting Up a Connection Pool Password with Java MBeans

Figure 7–3 shows the hierarchy of the configuration MBeans that you use to set up the password for connections to BRM applications. See "PCP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 7–3 BRM Connection Passwords Configuration MBeans



To set up the password:

1. Invoke the following operation of **CredentialStoreMBean**:

   ```
   void setPassword (string Key, string Password, boolean One-way)
   ```

   See Table 7–4 for more information about these parameters.

2. Invoke the following operation of **CredentialStoreMBean**:

   ```
   boolean validatePassword()
   ```

Managing Connection Pool Credentials

You can check whether a key exists in the Credential store and delete keys. You can manage keys using the Administration Console or Java MBeans.

Managing Credential Store Keys with the Administration Console

You can check whether a key exists in the Credential store and delete keys.

To check whether a key exists in the credential store:

1. In the navigation tree in the domain navigation pane, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Select **SSU PCP** node.
4. In the **PCP** tab, click the **Credential Store** tab.
5. In the **General** area, in the **Key** field, enter the key whose existence you want to check.
6. Click **Contains Key?**.

   The message which informs you whether the key exists, appears.
7. To close the message, click **OK**.

To delete a a specified key from the credential store:

- In the **Credential Store** tab, in the General area, in the Key field, enter the key and then click **Delete Key**.

To delete a all keys from the credential store:

- In the **Credential Store** tab, in the General area, click **Delete All Keys**.

Managing Keys in the Credential Store with Java MBeans

To manage keys in the credential store with Java MBeans, you use the same MBeans as described in "Setting Up a Connection Pool Password with Java MBeans".

To check whether a key exists in the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
boolean containsKey (string Key)
```

To delete a specified key from the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
void deleteKey (string Key)
```

To delete all keys from the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
void clear ()
```

Defining PCP Network Entities

You define PCP network entities using either the Administration Console or Java MBeans.

Defining PCP Network Entities with the Administration Console

To define a PCP network entity:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SSU PCP** node.

   The PCP SSU configuration pane appears. This pane displays a table listing PCP network entities. Each row represents one PCP network entity.
4. To define a new PCP network, at the bottom of the PCP SSU configuration pane, click the **New** button.

   The New dialog box appears.
5. Fill in the fields described in Table 7–5.

Table 7–5 PCP SSU Parameters

Field	Description
Name	A unique name that you give to the PCP network entity.

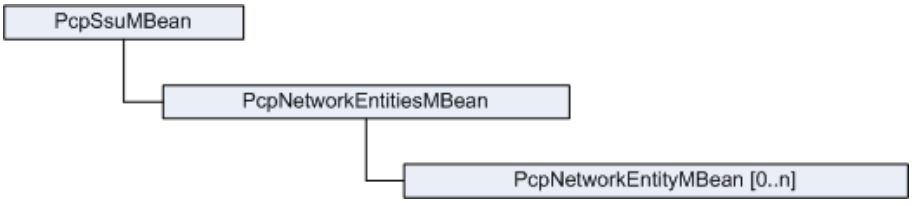
Table 7–5 (Cont.) PCP SSU Parameters

Field	Description
Alias	An alias that you assign to the PCP network entity. This alias is used by IMs to specify the destination of PCP requests. If two or more PCP network entities belong to the same BRM application, you assign the same alias to all of them. The PCP SSU distributes requests among network entities having the same alias. For information on how to use the alias in IMs, see ■ "Configuring Diameter Credit Control Application Parameters" in "Configuring IM-OCF PCP" of the *Oracle Communications Service Broker Processing Domain Configuration Guide* ■ "Configuring Call Handling Parameters" in "Configuring IM-OFCF PCP" of the *Oracle Communications Service Broker Processing Domain Configuration Guide*
Connection Pool ID	The identifier of the connection pool used to communicate with the remote PCP network entity. This should be one of the connection pools that you have previously defined. See "Defining Connection Pools"
Heartbeat	Specifies whether or not the PCP SSU activates the heartbeat mechanism over the connection with the network entity.
Response Timeout	The time interval, in seconds, during which the PCP SSU waits for a response from the network entity. If the PCP SSU does not receive a response within this time interval, then it considers the network entity inactive.
Active Interval	The time interval, in seconds, between two consecutive heartbeat requests sent to the network entity. This time interval is valid so long the network entity is considered active.
Inactive Interval	The time interval, in seconds, between two consecutive heartbeat requests sent to the network entity. This time interval is valid so long the network entity is considered inactive.

Defining PCP Network Entities with Java MBeans

Figure 7–4 shows the hierarchy of the configuration MBeans that you use to configure PCP network entities. See "PCP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 7–4 PCP SSU Configuration MBean Hierarchy



Each network entity is represented by PcpNetworkEntityMBean. You need to create a separate instance of PcpNetworkEntityMBean for each entity.

To create a network entity:

1. Create a new instance of **PcpNetworkEntityMBean** by invoking the following operation of the **PcpNetworkEntitiesMBean**:

```
ObjectName createPcpNetworkEntity(string NetworkEntityName)
```

2. Set the following attributes of the newly created instance of **PcpNetworkEntityMBean**:
   - Name
   - Alias
   - PcpPoolId
   - Heartbeat
   - ResponseTimeout
   - ActiveInterval
   - InactiveInterval

   See Table 7–5 for more information about these attributes.

PCP SSU Configuration MBeans Reference

The following sections provide reference information for the PCP SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

PcpSsuMBean

PcpSsuMBean is a root MBean for configuring the PCP SSU.

Object Name

com.convergin:Type=PcpSsu,Version=*MBean_Version*,Location=AdminServer,Name=ssupcp.ssupcp

Factory Method

Created automatically

Attributes

None

Operations

ObjectName getPcpNetworkEntities()

Returns a reference to the instance of PcpNetworkEntitiesMBean

PcpNetworkEntitiesMBean

PcpNetworkEntitiesMBean is the root MBean for defining BRM applications to which the PCP SSU routes a request. Using PcpNetworkEntitiesMBean, you need to create a separate instance of PcpNetworkEntityMBean for each BRM application.

Object Name

com.convergin:Type=PcpNetworkEntities,Version=*MBean_Version*,Location=AdminServer,Name=ssupcp.pcp_outbound_routing

Factory Method

Created automatically

Attributes

None

Operations

ObjectName getPcpNetworkEntity()

Returns a reference to the instance of PcpNetworkEntityMBean

ObjectName createPcpNetworkEntity()

Creates an instance of PcpNetworkEntityMBean

void destroyPcpNetworkEntity()

Destroys an instance of PcpNetworkEntityMBean

ObjectName lookupPcpNetworkEntity()

Searches for an instance of PcpNetworkEntityMBean

PcpNetworkEntityMBean

PcpNetworkEntityMBean enables you to define the BRM application to which the PCP SSU routes a request. You need to create a separate instance of PcpNetworkEntityMBean for each BRM application.

Object Name

com.convergin:Type=PcpNetworkEntity,Version=*MBean_Version*,Location=AdminServer,Name=ssupcp.*PcpNetworkEntityName*

Factory Method

Created automatically

Attributes

- Alias
- Heartbeat
- ResponseTimeout
- ActiveInterval
- InactiveInterval
- PcpPoolId

For more information about these attributes, see Table 7–5.

Operations

None

CredentialStoreMBean

CredentialStoreMBean enables you to set up passwords for BRM connections and the credential store. In addition, you can use CredentialStoreMBean to manage keys in the credential store.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=CredentialStore

Factory Method

Created automatically

Attributes

None

Operations

void setPassword()

Sets a password for the specified key

boolean validatePassword()

Validate a password against a password stored for the specified key

void setKeystore()

Validate a password against a password stored for the specified key

boolean containsKey()

Return true if the key exists in the credential store

void deleteKey()

Delete a key

void clear()

Remove all entries

pcpConfigMBean

pcpConfigMBean is the root MBean for configuring BRM connection pools.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig

Factory Method

Created automatically

Attributes

None

Operations

void addConnectionPools()

Add a new 'connectionPools' Element

void removeConnectionPools()

Add a new 'connectionPools' Element

void addOpcodeRetry()

Add a new 'opcodeRetry' Element

void removeOpcodeRetry()

Remove the 'opcodeRetry' Element

void addPcpTransactionIdleTimeout()

Add a new 'pcpTransactionIdleTimeout' Element

void removePcpTransactionIdleTimeout()

Remove the 'pcpTransactionIdleTimeout' Element

connectionPoolsMBean

connectionPoolsMBean is the root MBean for configuring connections with BRM applications.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools

Factory Method

pcpConfig.addConnectionPools()

Attributes

None

Operations

void addConnectionPoolConfig()

Add a new 'connectionPoolConfig' Element

void removeConnectionPoolConfig()

Remove the 'connectionPoolConfig' Element whose id is in connectionPoolConfig(...)

connectionPoolConfigMBean

connectionPoolConfigMBean represents a single connection with a BRM application.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*]

Factory Method

connectionPools.addConnectionPoolConfig()

Attributes

None

Operations

void cloneIt()

Clone the current element

void addCmPoolMaxSize()

Add a new 'cmPoolMaxSize' Element

void removeCmPoolMaxSize()

Remove the 'cmPoolMaxSize' Element

void addCmPoolMinSize()

Add a new 'cmPoolMinSize' Element

void removeCmPoolMinSize()

Remove the 'cmPoolMinSize' Element

void addCmPoolTimeout()

Add a new 'cmPoolTimeout' Element

void removeCmPoolTimeout()

Remove the 'cmPoolTimeout' Element

void addCmPoolMaxIdleTime()

Add a new 'cmPoolMaxIdleTime' Element

void removeCmPoolMaxIdleTime()

Remove the 'cmPoolMaxIdleTime' Element

void addCmPoolMaxRequestQueueSize()

Add a new 'cmPoolMaxRequestQueueSize' Element

void removeCmPoolMaxRequestQueueSize()

Remove the 'cmPoolMaxRequestQueueSize' Element

cmPoolIdMBean

cmPoolIdMBean enables you to specify the pool ID that you want to map to a physical connection.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolId

Factory Method

Created automatically

Attributes

cmPoolId

For more information about this attribute, see Table 7–2.

Operations

None

cmPoolMaxIdleTimeMBean

cmPoolMaxIdleTimeMBean enables you to specify the maximum idle time for a connection. When the specified time expires, the PSP SSU closes the connection.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolMaxIdleTime

Factory Method

connectionPoolConfig.addCmPoolMaxIdleTime()

Attributes

cmPoolMaxIdleTime

For more information about this attribute, see Table 7–2.

Operations

None

cmPoolMaxRequestQueueSizeMBean

cmPoolMaxRequestQueueSizeMBean enables you to specify the maximum size of the queue. When the maximum size is reached, the PCP SSU drops all further requests.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolMaxRequestQueueSize

Factory Method

connectionPoolConfig.addCmPoolMaxRequestQueueSize()

Attributes

cmPoolMaxRequestQueueSize

For more information about this attribute, see Table 7–2.

Operations

None

cmPoolMaxSizeMBean

cmPoolMaxSizeMBean enables you to specify the maximum number of connections in the pool.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolMaxSize

Factory Method

connectionPoolConfig.addCmPoolMaxSize()

Attributes

cmPoolMaxSize

For more information about this attribute, see Table 7–2.

Operations

None

cmPoolMinSizeMBean

cmPoolMinSizeMBean enables you to specify the minimum number of connections in the pool.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolMinSize

Factory Method

connectionPoolConfig.addCmPoolMinSize()

Attributes

cmPoolMinSize

For more information about this attribute, see Table 7–2.

Operations

None

cmPoolTimeoutMBean

cmPoolTimeoutMBean enables you to specify the time that the PCP SSU waits for establishing a connection.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=cmPoolTimeout

Factory Method

connectionPoolConfig.addcmPoolTimeout()

Attributes

cmPoolTimeout

For more information about this attribute, see Table 7–2.

Operations

None

enabledMBean

enabledMBean enables you to specify whether the connection is enabled.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=enabled

Factory Method

Created automatically

Attributes

enabled

For more information about this attribute, see Table 7–2.

Operations

None

hostMBean

hostMBean enables you to specify the IP address of the BRM application.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=host

Factory Method

Created automatically

Attributes

host

For more information about this attribute, see Table 7–2.

Operations

None

loginMBean

loginMBean enables you to specify the login that the PCP SSU uses to connect to the BRM application.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=login

Factory Method

Created automatically

Attributes

login

For more information about this attribute, see Table 7–2.

Operations

None

portMBean

portMBean enables you to specify the port of the BRM application.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=connectionPools,name2=connectionPoolConfig[*ConnectionPoolIndex*],name3=port

Factory Method

Created automatically

Attributes

port

For more information about this attribute, see Table 7–2.

Operations

None

opcodeRetryMBean

opcodeRetryMBean enables you to specify the number of attempts that the PCP SSU tries to execute a BRM opcode.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=opcodeRetry

Factory Method

pcpConfig.addOpcodeRetry()

Attributes

opcodeRetry

For more information about this attribute, see Table 7–1.

Operations

None

opcodeTimeoutMBean

opcodeTimeoutMBean enables you to specify the opcode timeout by default.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=opcodeTimeout

Factory Method

Created automatically

Attributes

opcodeTimeout

For more information about this attribute, see Table 7–1.

Operations

None

pcpTransactionIdleTimeoutMBean

pcpTransactionIdleTimeoutMBean enables you to specify the PCP transaction timeout.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.protocol_adapter.pcp.adapter,version=*MBean_Version*,name0=pcpConfig,name1=pcpTransactionIdleTimeout

Factory Method

pcpConfig.addPcpTransactionIdleTimeout()

Attributes

pcpTransactionIdleTimeout

For more information about this attribute, see Table 7–1.

Operations

None

8

Configuring SMPP Signaling Server Units

The following sections describe how to configure Service Broker SMPP SSUs using the Service Broker Administration Console and Java MBeans.

About the SMPP SSU

The SMPP SSU enables Service Broker to communicate with Short Message System Centers (SMSCs) through the Short Message Peer-to-Peer protocol.

When configuring the SMPP SSU, you set up the following:

- SMPP network entities, which define the SMSCs to which the SMPP SSU routes submit_sm messages generated by IM-UIX-SMS. See "Configuring SMPP Network Entities" for more information.
- SMSC connections, which define parameters of connections that the SMPP SSU establishes with SMSCs. See "Setting Up SMSC Connections" for more information.
- Incoming routing rules, which define the IM-UIX-SMS instance to which the SMPP SSU routes a delivery_sm message received from an SMSC. See "Configuring Incoming Routing Rules" for more information.

Configuring SMPP Network Entities

SMPP network entities are SMSCs to which the SMPP SSU routes submit_sm messages generated by IM-UIX-SMS. You set up rules that define the following:

- ID of the SMSCs to which the SMPP SSU routes the message
- Alias to be set in the IM-UIX-SMS configuration to route the message to the SMSC with a specified ID. To provide continuous operation in situations when an SMSC fails, you can map the same alias to multiple SMSCs. If one of the specified SMSCs fails, the SMPP SSU routes the message to another SMSC mapped to the same alias.
- Parameters of the heartbeat mechanism. Using this mechanism, the SMPP SSU regularly sends requests to an SMSC. If the SMPP SSU does not receive a response from the SMSC within the specified period of time, the SMPP SSU considers this SMSC inactive. The SMPP SSU does not send any further requests to this SMSC.

You can set up these parameters using the Administration Console and Java MBeans.

Configuring SMPP Network Entities with the Administration Console

To configure SMPP network entities:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP SSU** node.
4. In the SMPP SSU configuration pane, click the **SSU SMPP** tab and then the **SMPP Network Entities** subtab.

   The SMPP Network Entity configuration pane appears. This pane displays a table. The table contains rules that define to which SMSC the short message is routed. Each row in the table represents a single rule.
5. To create a new rule, at the bottom of the SMPP Network Entities configuration pane, click **New**.

   The New dialog box appears.
6. Fill in the fields described in Table 8–1.

Table 8–1 SMPP Network Entities Parameters

Field	Descriptions
Name	Specifies the name of the rule.
Alias	Specifies the SIP URI of the IM to which the SMPP SSU routes an incoming session. The alias has the following format: `SSU:`*IM-instance-name.IM-type@domain-id* ■ *IM-instance-name*: IM instance name you specified when you added this IM in the IM configuration pane. ■ *IM-type*: Type of IM instance. ■ *domain-id*: Name of the Processing Domain or Processing Domain Group where the relevant IM or application is deployed. This parameter is required only when your Service Broker deployment includes two or more Processing Domains. Use the name given to the domain when it was created. This name is specified by the axia.domain.id property. To set a Processing Domain Group, you must specify the group name. See "Managing Processing Domain Groups" in the *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information. Example: `ssu:im_uix-sms.IMUIXSMS@processing-domain.1` *domain-id* is required only if your deployment includes two or more Processing Domains. To provide continuous operation in situations when an SMSC fails, you can map the same alias to multiple SMSCs. If one of the specified SMSCs fails, the SMPP SSU routes the message to another SMSC mapped to the same alias.
Heartbeat	Specifies whether or not the SMPP SSU uses the heartbeat mechanism to regularly check whether the SMSC is active. Possible values: ■ ON ■ OFF Default value: ON

Table 8–1 (Cont.) SMPP Network Entities Parameters

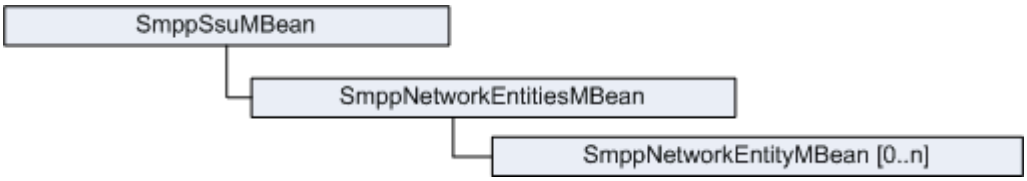
Field	Descriptions
SMSC Identifier	Specifies the ID of the SMSC to which the SMPP SSU routes the submit_sm message if the value of the Default SMSC Alias parameter set in the IM-UIX-SMS configuration and the value of the Alias parameter match.
Response Timeout	Specifies the time interval, in seconds, during which the SMPP SSU waits for a response from the SMSC. If the response timeout expires, and the SMPP SSU still does not receive a response, the SMPP SSU considers the SMSC inactive.
Active Interval	Specifies the time interval, in seconds, for sending heartbeat requests from the SMPP SSU to the SMSC. This field is used if the previous heartbeat test showed that the SMSC is active.
Inactive Interval	Specifies the time interval, in seconds, for sending heartbeat requests from the SMPP SSU to the SMSC. This field is used if the previous heartbeat test showed that the SMSC is inactive.

7. Click **Apply**.

Configuring SMPP Network Entities with Java MBeans

Figure 8–1 shows the hierarchy of the configuration MBeans that you use to configure SMPP network entities. See "SMPP SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 8–1 SMPP Network Entities Configuration MBeans



Each SMPP network entity is represented by SmppNetworkEntityMBean. You need to create a separate instance of SmppNetworkEntityMBean for each network entity.

To create an SMPP network entity:

1. Create an instance of SmppNetworkEntityMBean by invoking the following operation of SmppNetworkEntitiesMBean:

   ```
   ObjectName createSmppNetworkEntity(string EntityName)
   ```

2. Set the following attributes of the newly created instance of SmppNetworkEntityMBean:
   - Name
   - Alias
   - Heartbeat
   - SmscId
   - ResponseTimeout
   - ActiveInterval
   - InactiveInterval

See Table 8–1 for more information about these attributes.

Setting Up SMSC Connections

To route a submit_sm message to an SMSC, you set up connection between the SMPP SSU and SMSCs. Setting up a connection requires configuration of the following parameters:

- General parameters, which define parameters which are common for all connections to SMSCs.
- SMSC connection parameters, which define settings required for each connection. When setting up a connection, you map SMSC IDs specified in SMPP Network Entities, to physical addresses of SMSCs.
- Credential store parameters, which define settings for secure storage of connection passwords.

You can set up these parameters using the Administration Console and Java MBeans.

Configuring General Parameters

General parameters define parameters which are common for all connections to SMSCs. You can configure general parameters using the Administration Console or Java MBeans.

Configuring General Parameters with the Administration Console

To configure general parameters:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP** node.
4. In the SMPP configuration pane, click the **SMPP** tab and then the **General** subtab.
5. Fill in the fields described in Table 8–2.

Table 8–2 General Parameters

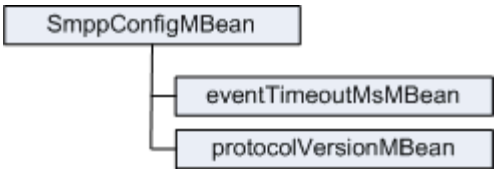
Field	Descriptions
protocolVersion	Specifies the version of the SMPP protocol that the SMPP SSU uses to communicate with SMSCs.
eventTimeoutMs	Specifies the timeout for an incoming event in milliseconds. Default value: 10000

6. Click **Apply**.

Configuring General Parameters with Java MBeans

Figure 8–2 shows the hierarchy of the configuration MBeans that you use to configure connections to SMSCs. See "SMPP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 8–2 General Parameters Configuration MBeans



To configure general parameters:

1. Set the **eventTimeoutMs** attribute of **eventTimeoutMsMBean** as described in Table 8–2.
2. Set the **protocolVersion** attribute of **protocolVersionMBean** as described in Table 8–2.

See "SMPP SSU Configuration MBeans Reference" for more information about these MBeans.

Setting Up Connection Pools

Connection parameters define settings required for each connection.When setting up a connection, you map SMSC IDs specified in SMPP Network Entities, to physical addresses of SMSCs.

You can configure connection parameters using the Administration Console or Java MBeans.

Setting Up Connection Pools with the Administration Console

To configure connection parameters:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP** node.
4. In the SMPP configuration pane, click the **SMPP** tab and then the **SMSC** subtab.
5. Fill in the fields described in Table 8–3.

Table 8–3 SMSc Connections Parameters

Field	Descriptions
SMSC Identifier	Specifies the ID of the SMSC for which you set up a connection. The value that you specify in this parameter must correspond to the SmscId parameter which you set in the SMPP Network Entities configuration.
SMSC Address	Specifies the host name or IP address of the SMSC to which the SMPP SSU routes a submit_sm message.
SMSC Port	Specifies the port of the SMSC to which the SMPP SSU routes a submit_sm message.
ESME System ID	Specifies the ID of the External Short Messaging Entity (ESME) that the SMPP SSU uses to bind to the SMSC.
ESME Credential Key	Specifies the key that the SMPP SSU uses to retrieve the ESME password from the credential store. See "Securing Connection Pools" for more information about specifying a password for a key.

Table 8–3 (Cont.) SMSc Connections Parameters

Field	Descriptions
ESME System Type	Specifies the type of the ESME system that the SMPP SSU uses to bind to the SMSC.
ESME Address Ton	Specifies the Type Of Number of the ESME address that the SMPP SSU uses to bind to the SMSC.
ESME Address NPI	Specifies the Numbering Plan Indicator of the ESME address that the SMPP SSU uses to bind to the SMSC.
ESME Address Range	Specifies the range of the ESME address that SMPP SSU uses to bind to the SMSC. Default value: .*
Local Address	Specifies the local address (hostname or IP) used to connect to the SMSC. To use any address, leave this parameter empty.
ESME Port	Specifies the local TCP port used to connect to the SMSC. Use -1 for any port. Default value: -1
Bind Type	Specifies the type of connection to the SMSC. Possible values: ■ TRANSCEIVER ■ TRANSMITTER ■ RECEIVER Default value: TRANSCEIVER
Connection Pool Size	Specifies the size of the connection pool. Default value: 1
Connection Timer (sec)	Specifies the time, in seconds, that the SMPP SSU waits between connection attempts to the SMSC. Default value: 30
Request Timeout (ms)	Specifies the period of time, in milliseconds, that the SMPP SSU waits to consider the request timed out. Default value: 10000
Enquire Link Timer (sec)	Specifies the frequency, in seconds, with which the SMPP SSU sends a Enquire Link PDU on each SMSC connection. To disable sending a Enquire Link PDU, enter 0. Default value: 30
Window Size	Specifies the maximum number of pending requests for each TCP connection. To disable limitation, enter 0. Default value: 0
Connection Acquire Timeout (ms)	Specifies the timeout, in milliseconds, that the SMPP SSU waits for an available connection when no connections are currently available. This parameter is applicable only when the value of the windowSize parameter is greater than 0. Default value: 1000
Target	Specifies the name of the managed server to which this configuration applies. If you leave this parameter empty, the configuration applies to all managed servers.

Setting Up Connection Pools with Java MBeans

Figure 8–3 shows the hierarchy of the configuration MBeans that you use to configure connections with SMSCs. See "SMPP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 8–3 SMSCs Connections Configuration MBeans

Each connection to an SMSC is represented by an instance of **SmscMBean**. You need to create a separate instance of **SmscMBean** for each connection. When you create an instance of **SmscMBean**, the set of child MBeans are created. Each child MBean represent a single parameter of the connection.

For example, **SmscIdMBean** has the SmscId attribute that represents the SMSC ID parameter. Similarly, **smscAddressMBean** has the smscAddress attribute that represents the SMSC Address parameter.

To configure connections to SMSCs:

1. Create an instance of **SmscMBean** by invoking the following operation of **SmppConfigMBean**:

   ```
   void addSMSC()
   ```

2. Set the attributes of the child MBeans as described in Table 8–4.

Table 8–4 SmscMBean Child MBeans and Their Attributes

MBean	Attribute
SMSCIdMBean	SMSCId
bindTypeMBean	bindType
connectTimerSecMBean	connectTimerSec
connectionAcquireTimeoutMsMBean	connectionAcquireTimeoutMs
connectionPoolSizeMBean	connectionPoolSize
enquireLinkTimerSecMBean	enquireLinkTimerSec
esmeAddressNpiMBean	esmeAddressNpi
esmeAddressRangeMBean	esmeAddressRange
esmeAddressTonMBean	esmeAddressTon
esmeCredentialKeyMBean	esmeCredentialKey
esmeSystemIdMBean	esmeSystemId
esmeSystemTypeMBean	esmeSystemType
localAddressMBean	localAddress
localPortMBean	localPort
requestTimeoutMsMBean	requestTimeoutMs
smscAddressMBean	smscAddress
smscPortMBean	smscPort
targetMBean	target
windowSizeMBean	windowSize

   See Table 8–3 for more information about these attributes.

Securing Connection Pools

When communicating with SMSCs, Service Broker acts as an External Short Messaging Entity (ESME). A connection between an ESME and SMSC can be established if the ESME provides a proper password. This password is stored in the credential store under a certain key. You specify this key by defining the ESME Credential Key parameter when setting up connection pools (for more information, see Table 8–3).

You must specify passwords for different keys using the Administration Console or Java MBeans.

Setting Up a Connection Pool Password with the Administration Console

To specify a password for a connection:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.

3. Select the **SMPP** node.
4. In the SMPP configuration pane, click the **SMPP** tab and then the **Credential Store** subtab.
5. In the **Password** area, fill in the fields described in Table 8–5:

Table 8–5 Password Fields

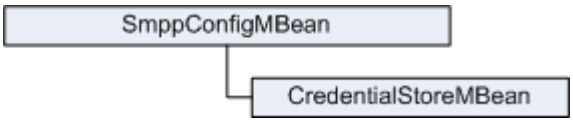
Field	Descriptions
Key	Specifies the key that the SMPP SSU uses to retrieve the ESME password from the credential store. You provide the key to be used for each connection pool. See "Setting Up Connection Pools" for more information.
Password	Specifies the password for the specified key.
One-way	Specifies how the SMPP SSU stores and retrieves the password. Do one of the following: ▪ If you want the SMPP SSU to include credentials in outgoing requests and thus, authenticate to an external entity, clear the **One-way** check box. ▪ If you want the SMPP SSU to include credentials in incoming requests only, select the **One-way** check box.

6. Click **Set Password**.

Setting Up a Connection Pool Password with Java MBeans

Figure 8–4 shows the hierarchy of the configuration MBeans that you use to specify the password for connections to SMSCs. See "SMPP SSU Configuration MBeans Reference" for the object names of these MBeans.

Figure 8–4 Credential Store Configuration MBeans



To set up the password:

1. Invoke the following operation of **CredentialStoreMBean**:

   ```
   void setPassword (string Key, string Password, boolean One-way)
   ```

   See Table 8–5 for more information about these parameters.
2. Invoke the following operation of **CredentialStoreMBean**:

   ```
   boolean validatePassword()
   ```

Setting Up the Key Store Password and URL

The SMPP SSU stores connections and their passwords in the encrypted file known as Credential Store. You specify the name and location of the Credential Store file using the Java system property **javax.net.ssl.keyStore**. See "Configuring Security", *Oracle Communications Server Broker System Administrator's Guide* for more information.

You can set up the key store password using the Administration Console or Java MBeans.

Setting Up the Key Store Password with the Administration Console

To set up the key store password:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP** node.
4. In the SMPP configuration pane, click the **SMPP** tab and then the **Credential Store** subtab.
5. In the **KeyStore** area, fill in the fields described in Table 8–6:

Table 8–6 KeyStore Fields

Field	Descriptions
Key	Specifies a key under which the keystore is stored.
KeyStore Password	Specifies the password that protects the keystore.
KeyStore URL	Specifies the URL of the keystore file on a hard drive.

Setting Up the KeyStore Password with Java MBeans

To set up the credential store password with Java MBeans, you use the same MBeans as described in "Setting Up a Connection Pool Password with Java MBeans".

To set up the key store password:

- Invoke the following operation of **CredentialStoreMBean**:

  ```
  void setKeystore (string Key, string KeystorePassword, string KeystoreURL)
  ```

  See Table 8–6 for more information about these parameters.

Managing Keys in the Credential Store

You can check whether a key exists in the Credential store and delete keys. You can manage keys using the Administration Console or Java MBeans.

Managing Keys in the Credential Store with the Administration Console

To check whether a key exists in the credential store:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP** node.
4. In the SMPP SSU configuration pane, click the **SMPP** tab and then the **Credential Store** subtab.
5. In the General area, in the Key field, enter the key whose existence you want to check.
6. Click **Contains Key?**.

   The message which informs you whether the key exists, appears.
7. To close the message, click **OK**.

To delete a specified key from the credential store:

- In the **Credential Store** tab, in the General area, in the Key field, enter the key and then click **Delete Key**

To delete all keys from the credential store:

- In the **Credential Store** tab, in the General area, click **Delete All Keys.**

Managing Keys in the Credential Store with Java MBeans

To manage keys in the credential store with Java MBeans, you use the same MBeans as described in "Setting Up a Connection Pool Password with Java MBeans".

To check whether a key exists in the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
boolean containsKey (string Key)
```

To delete a specified key from the credential store:

- Invoke the following operations of **CredentialStoreMBean**

```
void deleteKey (string Key)
```

To delete all keys from the credential store:

- Invoke the following operations of **CredentialStoreMBean**:

```
void clear ()
```

Configuring Incoming Routing Rules

Incoming routing rules define the IM-UIX-SMS instance to which the SMPP SSU routes a delivery_sm message received from the SMSC. For each rule, you define the following parameters:

- Conditions:
  - Destination address
  - Service Type
- Alias of the IM-UIX-SMS instance to which the SMPP SSU routes the message if both conditions are met

You can set up incoming routing rules using the Administration Console and Java MBeans.

Configuring Incoming Routing Rules with the Administration Console

To configure incoming routing rules:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Select the **SMPP SSU** node.

   The SMPP SSU configuration pane appears.
4. In the SMPP SSU configuration pane, click the **SSU SMPP** tab and then the **Incoming Routing Rules** subtab.

   The Incoming Routing Rules configuration pane appears. This pane displays a table. The table contains rules that define to which instance of IM-UIX-SMS the **delivery_sm** message is routed. Each row in the table represents a single rule.

5. To create a new rule, at the bottom of the Incoming Routing Rules configuration pane, click **New**.

   The New dialog box appears.

6. Fill in the fields described in Table 8–7.

Table 8–7 Incoming Routing Rules Parameters

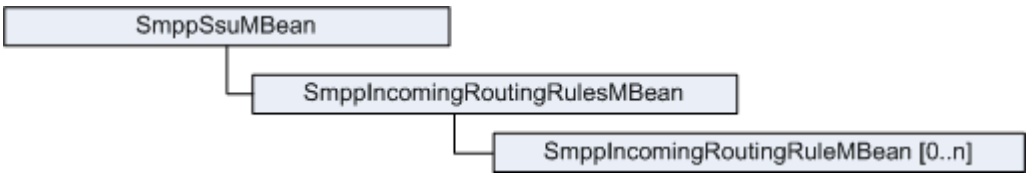
Field	Descriptions
Name	Specifies the name of the rule.
SMPP Destination Address	Specifies the destination address to be set in the delivery_sm message.
Service Type	Specifies the service type to be set in the **delivery_sm** message.
Alias	Specifies the alias of the IM-UIX-SMS instance. The SMPP SSU routes the delivery_sm message to this instance if the destination address and service type set in the delivery_sm message match the values set in SMPP Destination Address and Service Type parameters.

7. Click **Apply**.

Configuring Incoming Routing Rules with Java MBeans

Figure 8–5 shows the hierarchy of the configuration MBeans that you use to configure incoming routing rules. See "SMPP SSU Configuration MBeans Reference" for object names of these MBeans.

Figure 8–5 Incoming Routing Rules Configuration MBeans



Each incoming routing rule is represented by SmppIncomingRoutingRuleMBean.

You need to create a separate instance of SmppIncomingRoutingRuleMBean for each incoming routing rule.

To create an incoming routing rule:

1. Create an instance of SmppIncomingRoutingRuleMBean by invoking the following operation of SmppIncomingRoutingRulesMBean:

   ```
   ObjectName createSmppIncomingRoutingRule(string RuleName)
   ```

2. Set the following attributes of the newly created instance of SmppIncomingRoutingRuleMBean:

   - Name
   - SmppDestinationAddress
   - ServiceType
   - Alias

   See Table 8–7 for more information about these attributes.

SMPP SSU Configuration MBeans Reference

The following sections provide reference information for the SMPP SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

SmppSsuMBean

SmppSsuMBean is the root MBean for configuring SMPP SSU.

Object Name

com.convergin:Type=SmppSsu,Version=*MBean_Version*,Location=AdminServer,Name=ssusmpp.ssusmpp

Factory Method

Created automatically.

Attributes

None

Operations

ObjectName getSmppIncomingRoutingRules()

Returns a reference to the instance of SmppIncomingRoutingRulesMBean.

ObjectName getSmppNetworkEntities()

Returns a reference to the instance of SmppNetworkEntitiesMBean.

SmppIncomingRoutingRulesMBean

SmppIncomingRoutingRulesMBean is the root MBean for configuring SMPP incoming routing rules. These rules define the instance of IM-UIX-SMS to which the SMPP SSU routes a delivery_sm message received from the SMSC. Using SmppIncomingRoutingRulesMBean, you need to create a separate instance of SmppIncomingRoutingRuleMBean for each rule.

Object Name

com.convergin:Type=SmppIncomingRoutingRules,Version=*MBean_Version*,Location=AdminServer,Name=ssusmpp.smpp_inbound_routing

Factory Method

Created automatically.

Attributes

None

Operations

ObjectName getSmppIncomingRoutingRule()

Returns a reference to the instance of SmppIncomingRoutingRuleMBean

ObjectName createSmppIncomingRoutingRule()

Creates an instance of SmppIncomingRoutingRuleMBean

void destroySmppIncomingRoutingRule()

Destroys an instance of SmppIncomingRoutingRuleMBean

ObjectName lookupSmppIncomingRoutingRule()

Searches for an instance of SmppIncomingRoutingRuleMBean

SmppIncomingRoutingRuleMBean

SmppIncomingRoutingRuleMBean enables you to set up a single incoming routing rule. This rule defines the instance of IM-UIX-SMS to which the SMPP SSU routes a delivery_sm message received from the SMSC. You need to create a separate instance of SmppIncomingRoutingRuleMBean for each rule.

Object Name

com.convergin:Type=SmppIncomingRoutingRule,Version=*MBean_Version*,Location=AdminServer,Name=ssusmpp.*incoming_routing_rule_name*

Factory Method

SmppIncomingRoutingRules.createSmppIncomingRoutingRule()

Attributes

- Name
- SmppDestAddress
- ServiceType
- Alias

For more information about these attributes, see Table 8–7.

Operations

None

SmppNetworkEntitiesMBean

SmppNetworkEntitiesMBean is the root MBean for defining SMSCs to which the SMMP SSU routes a submit_sm message generated by IM-UIX-SMS. Using SmppNetworkEntitiesMBean, you need to create a separate instance of SmppNetworkEntityMBean for each SMSC.

Object Name

com.convergin:Type=SmppNetworkEntities,Version=*MBean_Version*,Location=AdminServer,Name=ssusmpp.smpp_outbound_routing

Factory Method

Created automatically.

Attributes

None

Operations

ObjectName getSmppNetworkEntity()

Returns a reference to the instance of SmppNetworkEntityMBean

ObjectName createSmppNetworkEntity()

Creates an instance of SmppNetworkEntityMBean

void destroySmppNetworkEntity()

Destroys an instance of SmppNetworkEntityMBean

ObjectName lookupSmppNetworkEntity()

Searches for an instance of SmppNetworkEntityMBean

SmppNetworkEntityMBean

SmppNetworkEntityMBean enables you to define the SMSC to which the SMPP SSU routes a submit_sm message generated by IM-UIX-SMS. You need to create a separate instance of SmppNetworkEntityMBean for each SMSC.

Object Name

com.convergin:Type=SmppNetworkEntity,Version=*MBean_Version*,Location=AdminServer,Name=ssusmpp.*network_entity_name*

Factory Method

SmppNetworkEntities.createSmppNetworkEntity()

Attributes

- Name
- Alias
- Heartbeat
- SmscId
- ResponseTimeout
- ActiveInterval
- InactiveInterval

For more information about these attributes, see Table 8–1.

Operations

None

SmppConfigMBean

SmppConfigMBean is the root MBean for configuration of connections that the SMPP SSU establishes with SMSCs. Using SmppConfigMBean, you need to create a separate instance of SmscMBean for each SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol

Factory Method

Created automatically.

Attributes

None

Operations

void addSMSC()

Creates a new instance of SmscMBean

void removeSMSC()

Removes an instance of SmscMBean

SmscMBean

SmscMBean enables you to configure a connection with a single SMSC. SmscMBean is created with a set of child MBeans. Each child MBean represents a configuration parameter of the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*]

Factory Method

SmppConfig.addSMSC()

Attributes

None

Operations

void CloneIt()

Clones the current instance of SmscMBean.

SMSCIdMBean

SMSCIdMBean enables you to specify the ID of the SMSC to which the SMPP SSU routes a submit_sm message.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*2.0.0*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=SMSCId

Factory Method

Created automatically.

Attributes

- SMSCId

See Table 8–3 for more information about this attribute.

Operations

None

bindTypeMBean

bindTypeMBean enables you to specify the type of connection to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=bindType

Factory Method

Created automatically.

Attributes

- BindType

See Table 8–3 for more information about this attribute.

Operations

None

connectTimerSecMBean

connectTimerSecMBean enables you to specify the time in seconds between each connection attempt to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=connectTimerSec

Factory Method

Created automatically.

Attributes

- ConnectTimerSec

See Table 8–3 for more information about this attribute.

Operations

None

connectionAcquireTimeoutMsMBean

connectionAcquireTimeoutMsMBean enables you to specify the timeout in milliseconds that the SMPP SSU waits for an available connection when no connections are currently available. This parameter is applicable only when the value of the windowSize parameter is greater than 0.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=connectionAcquireTimeoutMs

Factory Method

Created automatically.

Attributes

- ConnectionAcquireTimeoutMs

See Table 8–3 for more information about this attribute.

Operations

None

connectionPoolSizeMBean

connectionPoolSizeMBean enables you to specify the number of connections used towards the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=connectionPoolSize

Factory Method

Created automatically.

Attributes

- ConnectionPoolSize

See Table 8–3 for more information about this attribute.

Operations

None

enquireLinkTimerSecMBean

enquireLinkTimerSecMBean enables you to specify the frequency in seconds that Enquire Link PDU's are sent on each SMSC connection.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=enquireLinkTimerSec

Factory Method

Created automatically.

Attributes

- EnquireLinkTimerSec

See Table 8–3 for more information about this attribute.

Operations

None

esmeAddressNpiMBean

esmeAddressNpiMBean enables you to specify the ESME address Numbering Plan Indicator used when binding to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeAddressNpi

Factory Method

Created automatically.

Attributes

- EsmeAddressNpi

See Table 8–3 for more information about this attribute.

Operations

None

esmeAddressRangeMBean

esmeAddressRangeMBean enables you to specify the ESME address range used when binding to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeAddressRange

Factory Method

Created automatically.

Attributes

- EsmeAddressRange

See Table 8–3 for more information about this attribute.

Operations

None

esmeAddressTonMBean

esmeAddressTonMBean enables you to specify the ESME address Type Of Number used when binding to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeAddressTon

Factory Method

Created automatically.

Attributes

- EsmeAddressTon

See Table 8–3 for more information about this attribute.

Operations

None

esmeCredentialKeyMBean

esmeCredentialKeyMBean enables you to specify the key used to retrieve the ESME password from the credential store.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeCredentialKey

Factory Method

Created automatically.

Attributes

- EsmeCredentialKey

See Table 8–3 for more information about this attribute.

Operations

None

esmeSystemIdMBean

esmeSystemIdMBean enables you to specify the ESME system ID used when binding to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeSystemId

Factory Method

Created automatically.

Attributes

- EsmeSystemId

See Table 8–3 for more information about this attribute.

Operations

None

esmeSystemTypeMBean

esmeSystemTypeMBean enables you to specify the ESME system type used when binding to the SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=esmeSystemType

Factory Method

Created automatically.

Attributes

- EsmeSystemType

See Table 8–3 for more information about this attribute.

Operations

None

localAddressMBean

localAddressMBean enables you to specify the local address (hostname or IP) used to connect to SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=localAddress

Factory Method

Created automatically.

Attributes

- LocalAddress

See Table 8–3 for more information about this attribute.

Operations

None

localPortMBean

localPortMBean enables you to specify the local port used when connecting to SMSC.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=localPort

Factory Method

Created automatically.

Attributes

- LocalPort

See Table 8–3 for more information about this attribute.

Operations

None

requestTimeoutMsMBean

requestTimeoutMsMBean enables you to specify the number of milliseconds until a request is considered to have timed out.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=requestTimeoutMs

Factory Method

Created automatically.

Attributes

- RequestTimeoutMs

See Table 8–3 for more information about this attribute.

Operations

None

smscAddressMBean

smscAddressMBean enables you to specify the SMSC host name or IP address.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=smscAddress

Factory Method

Created automatically.

Attributes

- SmscAddress

See Table 8–3 for more information about this attribute.

Operations

None

smscPortMBean

smscPortMBean enables you to specify the SMSC port.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=smscPort

Factory Method

Created automatically.

Attributes

- SmscPort

See Table 8–3 for more information about this attribute.

Operations

None

targetMBean

targetMBean enables you to specify the target managed server this configuration applies to. If omitted, the configuration applies to all managed servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=target

Factory Method

Created automatically.

Attributes

- Target

See Table 8–3 for more information about this attribute.

Operations

None

windowSizeMBean

windowSizeMBean enables you to specify the allowed window of unacknowledged SMPP requests sent to the SMSC per connection.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.smpp.esme,version=*MBean_Version*,name0=smppConfig,name1=SMSC[*SMSC_index*],name2=windowSize

Factory Method

Created automatically.

Attributes

- WindowSize

See Table 8–3 for more information about this attribute.

Operations

None

EventTimeoutMsMBean

EventTimeoutMsMBean enables you to specify the number of milliseconds until inbound event is considered to have timed out.

Factory Method

Created automatically.

Attributes

- EventTimeoutMs

See Table 8–3 for more information about this attribute.

Operations

None

ProtocolVersionMBean

ProtocolVersionMBean enables you to specify the SMPP protocol version supported by SMPP Protocol Adapter.

Factory Method

Created automatically.

Attributes

- ProtocolVersion

See Table 8–3 for more information about this attribute.

Operations

None

9

Configuring the Web Services Signaling Server Unit

This chapter describes how to configure the Service Broker Web Services Signaling Server Unit (SSU) using the Service Broker Administration Console and Java MBeans.

About the Web Service SSU

The Web Services SSU enables Service Broker Processing Tier components to communicate with external entities using SOAP or REST over HTTP. Service Broker can act as a Web services server or client to external entities through the Web Services SSU.

In general, you control Web service traffic through the Service Broker Signaling Tier using the following Web Services SSU components:

- Incoming routing rules, which map a request URL addressed by an incoming request to an internal service or IM.
- Outgoing routing rules, which specify external Web services to which Service Broker sends service requests.
- HTTP network access points, which specify connection-level settings for the Web services communications, such as the port on which Service Broker listens for HTTP traffic and security settings for connections.
- Specific settings for SOAP-based or REST-based HTTP connections.

The Processing Tier components that rely on the Web Services SSU are the Web Services IM (WS-IM) and Service Broker applications, such as Top Up or Subscriber Provisioning. These applications expose SOAP APIs that clients use to configure and manage their services.

Enabling access to the SOAP APIs involves the following general configuration steps:

1. Opening an HTTP listening port in the Web Services SSU configuration.
2. Configuring SSL security or authentication requirements for the connection.
3. Configuring an incoming routing rule that directs incoming client requests to the Web Service endpoint within Service Broker.

This chapter provides information about configuring the Web Services SSU. For more information about a particular SOAP API, see the implementation guide applicable to the Service Broker application, such as *Service Broker Subscriber Data User's Guide*.

The following procedures describe how to perform a task in the Administration Console. Alternatively, you can perform the configuration task by accessing the

corresponding configuration MBean from any JMX MBean client. The following sections identify the configuration MBeans that contain the attributes that correspond to the GUI fields.

Configuring Incoming Routing Rules

You use the **Incoming Routing Rules** tab to define how the Web Services SSU routes incoming Web service messages to internal Service Broker IMs and other applications.

To configure incoming routing rules for Web service messages:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. In the **SSU WS** tab, click the **Incoming Routing Rules** tab.

   The Incoming Routing Rules pane contains a table in which each row represents one Web services endpoint.
6. Click the **New** button.

   The New dialog box appears.
7. In the dialog box, provide values for the fields listed in Table 9–1.

Table 9–1 Web Services Incoming Routing Rules Parameters

Field	Description
Name	A unique name for the routing rule.
Service Name	The service name of the Service Broker Web service endpoint. Setting this field to **Any** causes all incoming web service messages to be routed to the specified module instance. This value is generally used to create a rule that routes incoming Web services messages to a default IM in the absence of a more specific routing rule. For the incoming routing rules for Service Broker applications that expose SOAP APIs, use the following service names: ▪ **BalanceManagerService**: Use for the routing rule for the Top Up and Balance Manager API service. ▪ **SubscriberProvisioning**: Use for the routing rule for the Subscriber Provisioning API service.

Table 9–1 (Cont.) Web Services Incoming Routing Rules Parameters

Field	Description
Alias	A logical name that specifies the Service Broker IM or application to which the Web Services SSU routes an incoming Web services message. The format differs depending upon whether you are routing to an internal service or an IM. The alias has the following format for IMs: **ssu:***IM_instance_name.IM_type@domain_id* The alias has the following format for internal service: **ssu:***domain_Id/application_id* Where: ■ *IM_instance_name*: Name of the destination IM instance. This is the IM name you specified when you added this IM in the IM configuration pane. ■ *IM_type*: Type of the destination IM instance. ■ *domain-id*: Name of the Processing Domain or Processing Domain Group where the relevant IM or application is deployed. This parameter is required only when your Service Broker deployment includes two or more Processing Domains. Use the name given to the domain when it was created. This name is specified by the axia.domain.id property. To set a Processing Domain Group, you must specify the group name. See "Managing Processing Domain Groups" in the *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information. ■ *application_id*: Name of the destination Service Broker application. This is a static name assigned to each Service Broker application. This is **topup** for the Balance Manager API service and **provisioning** for the Subscriber Provisioning API service. For the Service Broker SOAP API services, use the following alias values (given the default domain ID of **ocsb**): ■ **ssu:ocsb/topup**: The alias for the Balance Manager API service. ■ **ssu:ocsb/provisioning**: The alias for the Subscriber Provisioning API service.

8. Click **OK** to save the new incoming routing rule configuration.

To create the routing rule through the MBean interface, invoke the WSIncomingRoutingRulesMBean.**createWSIncomingRoutingRule()** operation. You can then modify the attributes of the new WSIncomingRoutingRuleMBean instance, which correspond to the fields in Table 9–1.

Configuring Outgoing Routing Rules

You use the **Outgoing Routing Rules** tab to define how the Web Services SSU routes outgoing Web service messages to external Web service endpoints.

In the rule, you specify the address of each external Web service endpoint and assign an alias to each endpoint. IMs and Service Broker applications use the alias to refer to an external Web service destination.

To configure outgoing routing rules for Web service messages:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** tab.
5. In the **SSU WS** tab, click the **Outgoing Routing Rules** tab.

   The Outgoing Routing Rules pane contains a table in which each row represents one Web services endpoint.
6. Click the **New** button.

   The New dialog box appears.
7. In the dialog box, provide values for the fields listed in Table 9–2

Table 9–2 Web Services Outgoing Routing Rules Parameters

Field	Description
Name	A unique routing rule name.
Alias	A logical name that you assign to the Web services endpoint.
Web Service URI	The Uniform Resource Identifiers (URI) used to address the Web services endpoint. The format of the address is similar to Web Uniform Resource Locators (URLs). For example: **http://webservices.example.com/eventnotification**
Heartbeat	Whether the Web Services SSU periodically checks the Web services endpoint availability. Select **ON** to activate periodic availability check, or **OFF** to disable it.
Hearbeat Method	The HTTP method used to check endpoint availability. Service Broker supports the **GET** only. This field is ignored if you set the Heartbeat field to **OFF**.
Response Timeout	The amount of time, in seconds, Service Broker waits for a response from the Web services endpoint before the endpoint is considered unavailable. This field is ignored if you set the Heartbeat field to **OFF**.
Active Interval	The amount of time, in seconds, between consecutive endpoint availability checks if the last availability check showed that the endpoint was available.
Inactive Interval	The amount of time, in seconds, between consecutive endpoint availability checks if the last availability check showed that the endpoint was unavailable.

8. Click **OK** to save the new outgoing rule configuration.

To create the routing rule through the MBean interface, invoke the WSOutgoingRoutingRulesMBean.**createWSOutgoingRoutingRule()** operation. You can then modify the attributes of the new WSOutgoingRoutingRuleMBean instance, which correspond to the fields in Table 9–2.

Configuring HTTP Access Settings

To enable HTTP connections between Service Broker and external entities, you must configure the HTTP connection settings in the Web Services SSU.

The HTTP connection settings specify the port on which Service Broker listens for HTTP requests, timeout settings, security requirements, and general connection settings.

Configuring HTTP Server General Settings

The general HTTP server settings apply to connections to Service Broker through the Web Services SSU that are initiated by an external client.

To specify general timeout settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **HTTP** tab.

   The General configuration pane under the Server subtab appears.
6. Set the value of the **Timeout** field to the maximum number of milliseconds that Service Broker can use to process a request. If this time expires, Service Broker returns an error response to the client.

   Set to any value from 1000 and 60000. The default is 30000.
7. Click **Apply** to save your change.

To configure the general HTTP server settings through the MBean interface, use httpConfigMBean.serverConfigMBean.

Configuring HTTP Server Network Access Settings

The network access point specifies the port on which the Web Services SSU listens for HTTP traffic, including HTTP traffic in the form of SOAP and REST messages.

To configure HTTP server network access settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **HTTP** tab.
6. In the **Server** subtab, click the **Network Access** subtab.
7. Click the **New** button.

   The New dialog box appears.
8. In the dialog box, provide values for the fields listed in Table 9–3.

Table 9–3 HTTP Network Access Parameters

Field	Description
Server Address	The local IP address or hostname bound to this HTTP listener.
Server Port	An available port number on which the Signaling Server listens for HTTP traffic. Be sure to avoid entering a port number already in use by the system.

Table 9–3 (Cont.) HTTP Network Access Parameters

Field	Description
Protocol	The protocol used for the port, either **HTTP** or **HTTPS** for Secure HTTP.
SSL Client Auth	Whether to enable SSL client authentication for this access point. Set to **true** to enable SSL client authentication. Enable SSL client authentication only if this network access point uses HTTPS protocol. If enabled, clients attempting to connect to this access point must present a client certificate that matches one in the truststore. Set to **false** to disable SSL client authentication. For more information on security, see *Service Broker Administration Guide*.
Keystore ID	The identifier of the security keystore for the connection in the Credential Store. Only applicable if this network access point uses HTTPS. See *Service Broker Administration Guide* for more information on using the Credential Store.
Truststore ID	The identifier of the security truststore in the Credential Store. Only applicable if this network access point uses HTTPS. See *Service Broker Administration Guide* for more information on using the Credential Store.
Target	The target Signaling Server to which this configuration applies. If empty, it applies to all servers.

9. Click **OK** to save the new HTTP access configuration.

To create the HTTP network access point through the MBean interface, invoke the serverConfigMBean.**addNetworkAccessPoint()** operation. You can then modify the attributes of the new networkAccessPointMBean instance, which correspond to the fields in Table 9–3.

Creating or Modifying HTTP Server Security Contexts

You use the Security Context tab to apply authentication requirements to the resources exposed by Service Broker through the Web Services SSU. When authentication is required, Service Broker validates the credentials provided in incoming requests.

> **Note:** The HTTP server security context applies HTTP Basic Authentication requirements to requests. Alternatively, you can require credentials in the form of Web Service Security (WSSE) UsernameToken credentials. See "Authenticating SOAP Requests with WSSE UsernameToken Credentials" for more information.

You associate a security requirement to a resource by configuring a security context by URI path.

For instance, the default REST root URI context is exposed at **/rest**. If HTTP Basic Auth is enabled for this address, any resource available under the REST root URI (such as **/rest/subscriber**) has the same requirement, unless a more specific security context applies to it.

To configure a security context for HTTP access:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.

2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **HTTP** tab.
6. In the **Server** subtab, click the **Security Context** subtab.
7. In the Security Context pane, you can either:
   - Click **New** to create a new context.
   - Select an existing context in the list and click **Update** to modify its values.
8. In the dialog box, provide values for the fields listed in Table 9–4.

Table 9–4 HTTP Security Context Parameters

Field	Description
Context URI	The URI to which the security requirement applies.
Auth Method	The authentication method applied to the resource. Options are: ■ **NONE**: No authentication is required. ■ **BASIC**: HTTP Basic Authentication is required to access the resource.
Realm	The security realm value to be presented to clients who do not provide credentials.
Username	The required user name to be included in the requests.
Credential Key	A key that identifies the credential in the Credential Store. This key is a name for the credential provided when loading the password associated with the user in the Credential Store. See *Service Broker Administration Guide* for more information on the Credential Store.

9. Click **OK** to save the new security context configuration.

To create the security context through the MBean interface, invoke the serverConfigMBean.**addSecurityContext()** operation. You can then modify the attributes of the new securityContextMBean instance, which correspond to the fields in Table 9–4.

Configuring HTTP Client Settings

The HTTP client settings apply to outgoing connections. In this case, Service Broker acts as a client to external HTTP servers through the Web Services SSU.

To configure HTTP client settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **HTTP** tab.
6. Click the **Client** tab.

7. Modify, if required, the default settings for the outgoing connection listed in Table 9–5.

Table 9–5 HTTP Client Parameters

Field	Description
Connect Timeout	The amount of time, in milliseconds, Service Broker allows for establishing an HTTP connection to a remote server. If the timeout expires before receiving data, the connect attempt is abandoned. The default value is 50000 milliseconds. Value must be from 1000 to 60000.
Read Timeout	The amount of time, in milliseconds, Service Broker allows for reading data from a remote server on the established connection. If the timeout expires, the read attempt is aborted. The default value is 30000 milliseconds. Value must be from 1000 to 60000.

8. Click **Apply** to save your changes to the configuration.

To configure the HTTP client settings through the MBean interface, use httpConfigMBean.clientConfigMBean.

Configuring SOAP Web Service Access

As an HTTP-based protocol, SOAP is subject to the common HTTP connection settings configured in the HTTP tab. These include, for example, the port on which Service Broker listens for HTTP traffic, Basic Authentication security, and so on. See "Configuring HTTP Access Settings" for information on configuring common HTTP access settings.

In addition, you can configure specific settings that apply to SOAP-based communication with external SOAP clients or servers.

Configuring SOAP Server Settings

The SOAP server settings apply to client connections to Service Broker in which Service Broker acts as the Web Service provider or server front-end. These include connections made to the Subscriber Profile API and Balance Manager API services.

To enable the SOAP services, you must configure HTTP access settings. You can then configure specific settings for SOAP access as described in this section.

Configuring Common SOAP Server Settings

To configure general SOAP access settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **SOAP** tab.

   The Server settings pane appears.
6. Verify and, if required, modify the default settings listed in Table 9–6.

Table 9–6 SOAP Server Parameters

Field	Description
Root URI	The path root for the SOAP API services provided by Service Broker. The default is **/soap**. Together with the service location, this root path forms the complete URI for accessing the SOAP resource at the Service Broker Signaling Domain. For example, given the default path for the root URI and Subscriber Provisioning service, the full path would be: https://*hostname*:*port*/soap/SubscriberProvisioning
Timeout	The maximum amount of time, in milliseconds, Service Broker may take to generate a response before returning an error response to the client. The default value is 10000 milliseconds. Values can be from 1000 to 60000.

7. Click **Apply** to save your changes to the configuration.

To configure the SOAP server settings through the MBean interface, use wsSoapConfigMBean.serverConfigMBean.

Configuring the URI Path for a Specific SOAP Service

To view or change the URI path of a SOAP service, follow these steps:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Select the name of the service for which you want to modify the existing URI path, either **Subscriber Provisioning** or **Balance Manager**.
5. In the **End Point** tab, click the URI context for the service.
6. Click the **Update** button.
7. Set the **URI** field to the value of the new path. The path value should be preceded by a forward slash character, as in the default value.

   By default, this is **/SubscriberProvisioning** for the Subscriber Provisioning SOAP service, and **/BalanceManagerService** for the Balance Manager SOAP service. Together with the root URI path, this path makes up the URI at which clients address the SOAP service.
8. Click **OK**.

See "Authenticating SOAP Requests with WSSE UsernameToken Credentials" for information on configuring the authentication fields for the SOAP service.

To configure the Web service paths through the MBean interface, use either endpointMBean (for Subscriber Store Web Service) or endpointMBean (for Top Up Web Service).

Configuring SOAP Client Parameters

The SOAP client settings apply to outgoing connections. In this case, Service Broker acts as a client to external SOAP Web service providers.

To configure SOAP client settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **SOAP** tab.
6. Click the **Client** subtab.
7. Enter values for the fields listed in Table 9–7.

Table 9–7 SOAP Client Parameters

Field	Description
Connect Timeout	The amount of time, in milliseconds, Service Broker allows for establishing a connection to a remote server. If the timeout expires before Service Broker establishes the connection, the connect attempt is abandoned. The default value is **5000**. Values can be from 1000 to 60000. To disable time outs, set this value to **0**.
Read Timeout	The amount of time, in milliseconds, Service Broker allows for reading data from a remote server on an established connection. If the timeout expires, the read attempt is aborted. The default value is **30000**. Values can be from 1000 to 60000. To disable time outs, set the value to **0**.

8. Click **Apply** to save your changes to the configuration.

To configure the SOAP client settings through the MBean interface, use wsSoapConfigMBean.clientConfigMBean.

Authenticating SOAP Requests with WSSE UsernameToken Credentials

Service Broker can authenticate incoming SOAP requests that contain WSSE UsernameToken credentials, as specified by OASIS UsernameToken Profile 1.0. For general information on WSSE UsernameToken, see the OASIS Web Service Security specifications at:

`http://www.oasis-open.org/`

You enable WSSE UsernameToken credential requirement by SOAP service. That is, it can be enabled for the Subscriber Provisioning service and disabled for the Balance Manager service, for example.

A service that requires WSSE UsernameToken authentication should not be configured to require an HTTP Basic Authentication credential as well. See "Creating or Modifying HTTP Server Security Contexts" for more information about HTTP Basic Authentication security contexts.

Service Broker validates the WSSE UsernameToken credential against credentials stored in the Service Broker Credential Store. Before configuring service authentication as described below, add the credential to be authenticated to the Credential Store. See *Service Broker Administration Guide* for information about the Credential Store.

To apply WSSE UsernameToken credential authentication to incoming SOAP service requests, follow these steps:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **SOAP** tab.
6. Click the **Credential Store** subtab.
7. In the **Key** field, enter an alias for the credential.
8. In the **Password** field, enter the password value for the credential.
9. Verify that HTTP Basic Authentication is disabled for the underlying HTTP security context for the SOAP service as follows:
   a. Click the **HTTP** tab under the SSU Web Services node.
   b. In the **Server** subtab, click the **Security Context** subtab.
   c. Verify that the **Auth Method** value is **NONE** for the Context URI path applicable to the Web service. By default, the context path for all SOAP Web services exposed by Service Broker is **/soap**.
   d. If necessary, select the security context item and click the **Update** button to change the Auth Method.
10. Under the **OCSB** navigation tree, expand, if necessary, the **Signaling Tier** node and then the **SSU Web Services** node.
11. Click the name of the service for which you want to require WSSE UsernameToken authentication, either **BALANCE MANAGER** or **SUBSCRIBER PROVISIONING**.
12. In the End Point tab, click on the URI context for the service.
13. Click the **Update** button.
14. For the **Authentication Method** value, choose **USERNAME_TOKEN**.
15. For the **Username** value, enter the user name portion of the credential to be authenticated by WSSE UsernameToken authentication.
16. For the **Credential Key** value, enter the credential alias you used when storing the password to be validated into the Credential Store.
17. Click **OK** to save your changes to the configuration.

Clients of the service must submit valid WSSE UsernameToken credentials with their service requests.

To configure the SOAP authentication for the services through the MBean interface, use the interface of endpointMBean (for Subscriber Store Web Service) or endpointMBean (for Top Up Web Service).

Configuring REST Web Service Access

As an HTTP-based protocol, REST-based communication is subject to the common HTTP connection settings configured in the HTTP tab. These include, for example, the port on which Service Broker listens for HTTP traffic, Basic Authentication security, and so on. See "Configuring HTTP Access Settings" for information on configuring common HTTP access settings.

In addition, you can configure specific settings that apply to REST-based communication with external REST clients or servers.

Configuring REST Server Parameters

The REST server settings apply to client connections made to Service Broker in which Service Broker acts as the server or server front-end for a REST API service.

To configure REST server settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **REST** tab.
6. In the **Server** tab, verify and, if required, modify the default settings listed in Table 9–8.

Table 9–8 REST Server Parameters

Field	Description
Root URI	The URI path at which Service Broker exposes REST APIs. This path value forms the root of the address that clients use to access REST resources. For example, given the default path of **/rest**, the full address of a REST resource would be: https://*hostname*:*port*/rest/subscriber/carol
Timeout	The amount of idle time, in milliseconds, after which Service Broker releases a client connection on which it is awaiting data. The default value is 10000 milliseconds. Values can be from 1000 to 60000. To disable timeout, set the timeout to **0**.

7. Click **Apply** to save your changes to the configuration.

To configure the REST server settings through the MBean interface, use wsRestConfigMBean.clientConfigMBean.

Configuring REST Client Parameters

The REST client settings apply to outgoing connections. In this case, Service Broker acts as a client to external REST Web service providers.

To configure REST client settings:

1. In the navigation tree in the domain navigation pane, expand **OCSB**.
2. Expand **Signaling Tier**.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **REST** tab.
6. Click the **Client** subtab.
7. Verify and, if required, modify the default settings listed in Table 9–9.

Table 9–9 REST Client Parameters

Field	Description
Connect Timeout	The amount of time, in milliseconds, Service Broker allows to establish a connection to a remote server. If the timeout expires before receiving data, the connection attempt is abandoned. The default value is **5000**. Values can be from 1000 to 60000. To disable timeout, set the timeout to **0**.
Read Timeout	The amount of time, in milliseconds, Service Broker allows for reading data from a remote server on an established connection. If the timeout expires, the read attempt is abandoned. The default value is 30000 milliseconds. Values can be from 1000 to 60000. To disable timeout, set the timeout to **0**.

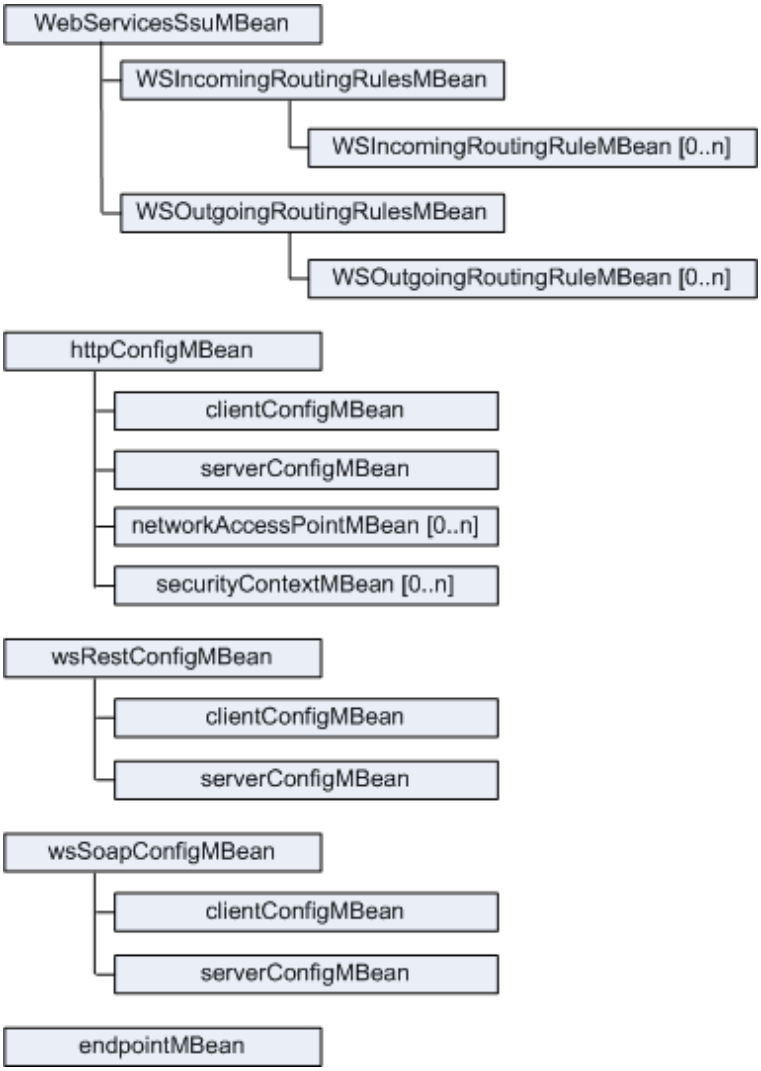
8. Click **Apply** to save your changes to the configuration.

To configure the REST client settings through the MBean interface, use wsRestConfigMBean.serverConfigMBean.

Web Services SSU Configuration MBeans Reference

Figure 9–1 shows the hierarchy of the configuration MBeans that you use to configure the Web Services SSU.

Figure 9–1 Web Services SSU and Related Configuration MBeans



The following sections provide reference information for the Web Services SSU configuration MBeans.

Note: MBeans described in this guide may include additional advanced attributes, which are not described in the guide. Advanced attributes are reserved for internal use. Do not change these attributes.

WebServicesSsuMBean

The root MBean for the Web Services SSU configuration.

Object Name

com.convergin:Type=WebServicesSsu,Version=2.0.0.0,Location=AdminServer,Name=ssuws.ssuws

Factory Method

Created automatically

Attributes

- **WSIncomingRoutingRules**

  A reference to the instance of **WSIncomingRoutingRules**.

- **WSOutgoingRoutingRules**

  A reference to the instance of **WSOutgoingRoutingRules**.

Operations

None

WSIncomingRoutingRulesMBean

A container for instances of **WsIncomingRoutingRulesMBean**, which represent the Web Services SSU inbound routing rule configuration.

Object Name

com.convergin:Type=WSIncomingRoutingRules,Version=2.0.0.0,Location=AdminServer,Name=ssuws.ws_inbound_routing

Factory Method

Created automatically

Attributes

- **WSIncomingRoutingRules**

The list of **WSIncomingRoutingRuleMBean** instances.

Operations

createWSIncomingRoutingRule()

Creates a new instance of **WSIncomingRoutingRuleMBean**.

destroyWSIncomingRoutingRule()

Removes an existing instance of **WSIncomingRoutingRuleMBean**.

lookupWSIncomingRoutingRule()

Searches for an instance of **WSIncomingRoutingRuleMBean**.

WSIncomingRoutingRuleMBean

An instance of an incoming routing rule. Each rule specifies how to route inbound Web services messages to a Service Broker IM or internal application.

For each rule, you specify a service name and the address alias of the internal application or IM destination.

Object Name

com.convergin:Type=WSIncomingRoutingRule,Version=2.0.0.0,Location=AdminServer,Name=ssuws.*instance_name*

Factory Method

WSIncomingRoutingRulesMBean.createWSIncomingRoutingRule()

Attributes

- **ServiceName**
- **Alias**

See Table 9–1 for information about these attributes.

WSOutgoingRoutingRulesMBean

A container for instances of WSOutgoingRoutingRuleMBean, which represent Web Services SSU outbound routing rules.

Object Name

com.convergin:Type=WSOutgoingRoutingRules,Version=2.0.0.0,Location=AdminServer,Name=ssuws.ws_outbound_routing

Factory Method

Created automatically

Attributes

- **WSOutgoingRoutingRules**

The list of WSOutgoingRoutingRuleMBean instances.

Operations

createWSOutgoingRoutingRule()

Creates a new instance of WSOutgoingRoutingRuleMBean.

destroyWSOutgoingRoutingRule()

Removes an existing instance of WSOutgoingRoutingRuleMBean.

lookupWSOutgoingRoutingRule()

Searches for an instance of WSOutgoingRoutingRuleMBean.

WSOutgoingRoutingRuleMBean

An instance of an outgoing routing rule. Each rule specifies how to route outbound Web services messages to external Web service end points.

For each rule, you specify an address, an alias, and how to check the availability of the endpoint.

Object Name

com.convergin:Type=WSOutgoingRoutingRule,Version=2.0.0.0,Location=AdminServer,Name=ssuws.*instance_name*

Factory Method

WSOutgoingRoutingRulesMBean.createWSOutgoingRoutingRule

Attributes

- **Alias**
- **WSUri**
- **Heartbeat**
- **HeartbeatMethod**
- **ResponseTimeout**
- **ActiveInterval**
- **InactiveInterval**

See Table 9–2 for information about these attributes.

httpConfigMBean

A container for instances of clientConfigMBean and serverConfigMBean, which provide common settings applicable to HTTP connections for the Web Services SSU.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.http,version=2.0.0,name0=httpConfig

Factory Method

Created automatically

clientConfigMBean

The HTTP client settings for the Web Services SSU. The settings apply to HTTP connections initiated by Service Broker to external HTTP servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.http,version=2.0.0,name0=httpConfig,name1=clientConfig

Factory Method

Created automatically

Attributes

- **connectTimeoutMs**
- **readTimeoutMs**

See Table 9–5 for information about these attributes.

serverConfigMBean

The HTTP server settings for the Web Services SSU. The settings apply to HTTP connections made to Service Broker by HTTP clients.

This MBean also acts as a container for instances of networkAccessPointMBean and securityContextMBean.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.http,version=2.0.0,name0=httpConfig,name1=serverConfig

Factory Method

Created automatically

Attributes

- **timeoutMs**

See "Configuring HTTP Server General Settings" for information about this attribute.

Operations

addNetworkAccessPoint()

Creates a new networkAccessPoint MBean.

removeNetworkAccessPoint()

Removes an existing networkAccessPoint MBean.

addSecurityContext()

Creates a new HTTP securityContext MBean instance.

removeSecurityContext()

Removes an existing securityContext MBean instance.

networkAccessPointMBean

An instance of an HTTP network access point. The network access point defines port number on which Service Broker listens for HTTP traffic.

For each access point, you specify the protocol type (HTTP or HTTPS), listen port number, and any SSL-level security requirements for connections.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.http,version=2.0.0,name0=httpConfig,name1=serverConfig,name2=networkAccessPoint[*instance_number*]

Factory Method

httpConfig.serverConfig.addNetworkAccessPoint()

Attributes

- **address**
- **port**
- **protocol**
- **clientAuthEnabled**
- **keystoreId**
- **truststoreId**
- **target**

See Table 9–3 for information about these attributes.

securityContextMBean

An instance of an HTTP security context. The security context defines HTTP-level client authentication requirements for accessing resources by URI.

The HTTP-level client authentication mechanisms are SSL client certificate authentication and HTTP Basic Authentication. See "serverConfigMBean" for information on configuring an authentication requirement based on WSSE UsernameToken credentials.

For each security context, you specify the protocol type (HTTP or HTTPS), listen port number, and any SSL-level security requirements for connections.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.http,version=2.0.0,name0=httpConfig,name1=serverConfig,name2=securityContext[*instance_number*]

Factory Method

Created automatically

Attributes

- **contextUri**
- **authMethod**
- **realm**
- **username**
- **credentialKey**

See Table 9–4 for information about these attributes.

wsSoapConfigMBean

The root MBean for the settings that apply to HTTP connections used to pass REST traffic between Service Broker and external clients or servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.soap,version=2.0.0,name0=wsSoapConfig

Factory Method

Created automatically

clientConfigMBean

The SOAP client settings for the Web Services SSU. The settings apply to HTTP connections used to pass SOAP traffic between Service Broker and external servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.soap,version=2.0.0,name0=wsSoapConfig,name1=clientConfig

Factory Method

Created automatically

Attributes

- **connectTimeoutMs**
- **readTimeoutMs**

See Table 9–7 for information about these attributes.

serverConfigMBean

The SOAP server settings for the Web Services SSU. The settings apply to HTTP connections used to pass SOAP traffic between Service Broker and external clients.

Note that to configure WSSE UsernameToken authentication requirements, set a credential in the Credential Store by using the following MBean:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.soap,version=2.0.0,name0=CredentialStore

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.soap,version=2.0.0,name0=wsSoapConfig,name1=serverConfig

Factory Method

Created automatically

Attributes

- **rootUri**
- **timeoutMs**

See Table 9–6 for information about these attributes.

wsRestConfigMBean

The root MBean for the settings that apply to HTTP connections used to pass REST traffic between Service Broker and external clients or servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.rest,version=2.0.0,name0=wsRestConfig

Factory Method

Created automatically

clientConfigMBean

The REST client settings for the Web Services SSU. The settings apply to HTTP connections used to pass REST traffic between Service Broker and external servers.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.rest,version=2.0.0,name0=wsRestConfig,name1=clientConfig

Factory Method

Created automatically

Attributes

- **connectTimeoutMs**
- **readTimeoutMs**

See Table 9–9 for information about these attributes.

serverConfigMBean

The REST server settings for the Web Services SSU. The settings apply to HTTP connections used to pass REST traffic between Service Broker and external clients.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.ws.rest,version=2.0.0,name0=wsRestConfig,name1=serverConfig

Factory Method

Created automatically

Attributes

- **rootUri**
- **timeoutMs**

See Table 9–8 for information about the attributes.

endpointMBean (for Subscriber Store Web Service)

An automatically created instance MBean that specifies the URI path on which the Web Services SSU exposes the Subscriber Store SOAP API.

For information on using the Subscriber Store API, see *Service Broker Subscriber Data User's Guide*.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.service.subscriber_store.provisioning.soap_ws,version=6.0.0.SNAPSHOT,
name0=webServiceSoap,name1=endpoint[0]

Factory Method

oracle.ocsb.app.rcc.service.subscriber_store.provisioning.soap_ws.webServiceSoap.addEndpoint()

Attributes

- **uri**
- **implementationClass**
- **authMethod**
- **username**
- **credentialKey**

See Table 9–8 for information about the attribute.

endpointMBean (for Top Up Web Service)

An automatically created instance MBean that specifies the URI path on which the Web Services SSU exposes the Top Up SOAP API.

For information on using the Top Up API, see *Service Broker Online Mediation Implementation Guide*.

Object Name

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.feature.topup.ws.soap_ws,version=6.0.0.SNAPSHOT,name0=webServiceSoap,name1=endpoint[0]

Factory Method

oracle.ocsb.app.rcc.feature.topup.ws.soap_ws.webServiceSoap.addEndpoint()

Attributes

- **uri**
- **implementationClass**
- **authMethod**
- **username**
- **credentialKey**

See Table 9–8 for information about the attribute.