Oracle® Communications Service Controller

Implementation Guide

Release 6.2

F18716-02

April 2020

Oracle Communications Service Controller Implementation Guide, Release 6.2

F18716-02

Contents

Preface

This document describes how to install, configure and use Oracle Communications Service Controller.

Audience

This document is intended for system integrators and system administrators who will install, configure, or administer Service Controller. It is also intended for developers that want to implement SIP applications on top of Service Controller, and require basic understanding of the Service Controller concepts.

This document assumes that you are already familiar with:

- Service Controllering strategies and standards
- Telecom networks, telecom protocols, and telecom standards, especially SIP and SS7-based protocols
- Service Controller concepts. For more information about concepts that are common to all products in the Service Controller product line, see *Service Controller Concepts Guide*
- The operating system on which your system is installed
- Java Management Extensions (JMX) Management Beans (MBeans)

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 that have purchased support 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

Service Controller Overview

This chapter provides an overview of the Oracle Communications Service Controller.

Before you read this chapter, you should be familiar with Service Controller concepts and architecture. See *Service Controller Concepts Guide* for information about concepts that are common to all products in the Service Controller product line.

About Service Controller

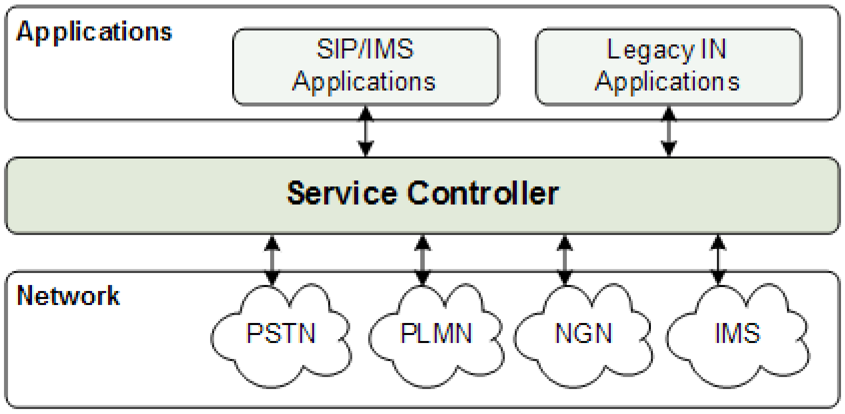
Service Controller is a core network element providing Service Controllering in IMS/LTE and traditional SS7 networks. Positioned between applications and session control entities, Service Controller controls service delivery for sessions executed in the network. Service Controller provides Intelligent Network (IN) applications and Session Initiation Protocol (SIP) applications, with connectivity and access to the network, and control of sessions running in the network.

Service Controller implements two major functions:

- Protocol mediation: Service Controller mediates between protocols, supporting various SS7 protocol variants and SIP, therefore enabling service delivery to different types of networks. By using Service Controller, you can deliver SIP services to subscribers in your traditional SS7 network, and IN Service Control Point (SCP) services to subscribers in your IMS/LTE network.
- Service Orchestration: Service Controller can invoke one or more applications, combining and delivering multiple services to sessions in the network. Service Controller supports mixed orchestration of SIP-based and IN-based applications, using an orchestration logic which defines how to route sessions through a number of applications; Service Controller invokes the applications in a particular order, according to conditions in the orchestration logic that determine which applications to invoke and in what order.

Figure 1–1 shows how Service Controller is positioned between applications and session control entities, providing applications with access to sessions in the network, mediating protocols, and orchestrating services.

Figure 1–1 Service Orchestration and Protocol Mediation Across Legacy SS7 Networks and IMS/LTE Networks



Service Controller Use Cases

With Service Controller you can implement various solutions:

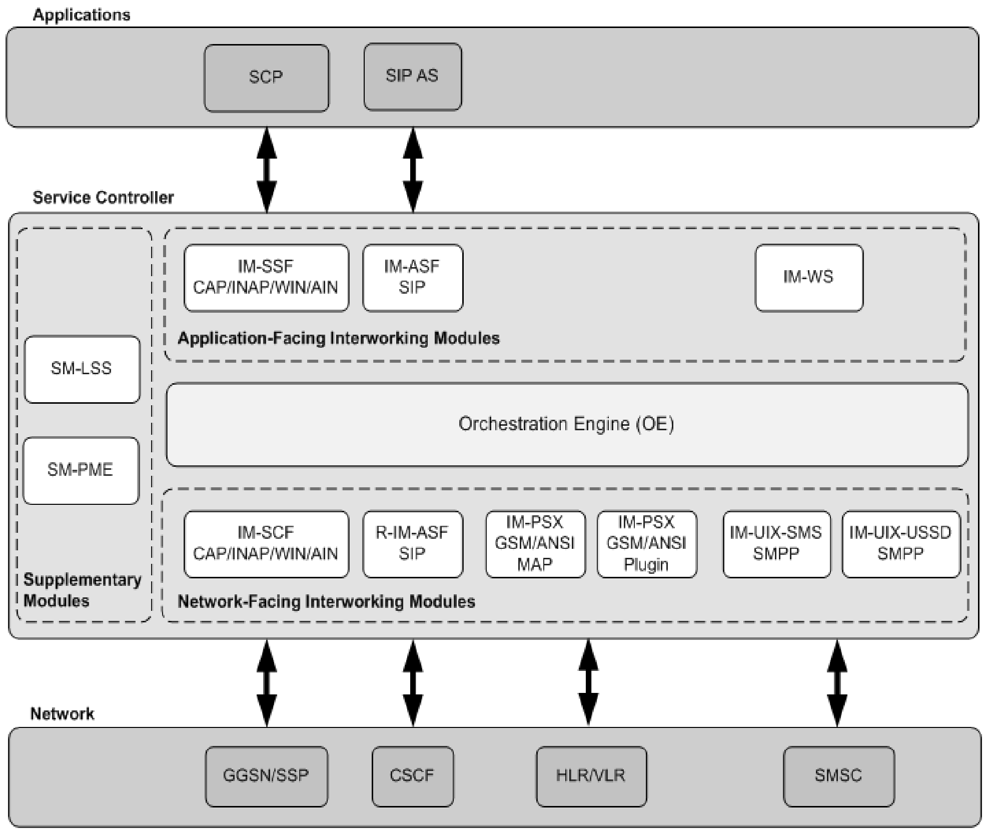
- Next Generation Intelligent Networks (NGIN): Delivery of combined SIP services to sessions in the legacy SS7-based network.
- Service Capability Interaction Manager (SCIM): Delivery of combined SIP services to sessions in the IMS network.
- Service orchestration: Delivery of combined legacy IN services with new SIP services, to sessions in the IMS/LTE network.
- IN mediation: Delivery of legacy IN services to sessions in the legacy SS7-based network, where the services and network use a different protocol variant. In this solution Service Controller converts between the two protocol variants.

For more information on the various solutions and how to implement them using Service Controller, see the solution specific chapters in this book.

The Service Controller Functional Architecture

The Service Controller signaling tier composes only Signaling Server Units (SSUs). The Service Controller processing tier includes interworking modules, supplementary modules and the Orchestration Engine, as shown in Figure 1–2.

Figure 1–2 Service Controller Functional Components in the Processing Tier



Inside Service Controller, communication between the Orchestration Engine and modules is normalized, based on the Session Abstraction Layer (SAL) protocol. Each module provides the conversion between the Service Controller internal SAL representation of the session and the applicable external protocol.

Orchestration Engine

The Orchestration Engine (OE) resides at the center of the Service Controller architecture and implements the service orchestration functionality. Sessions arriving through interworking modules invoke the Orchestration Engine which routes the sessions through one or more applications, based on an orchestration logic that the Orchestration Engine obtains for each session from the subscriber profile store (LSS or HSS). The Orchestration Engine invokes applications in a specific order, according to conditions that determine which application to invoke and in which order. See *Service Controller Orchestration User's Guide* for more information about application orchestration.

Interworking Modules

Interworking modules, often referred to as modules, provide the Orchestration Engine with communication to applications and session control entities in the SS7-based and IMS/LTE networks. Interworking modules implement the protocol mediation function; they convert messages of the relevant protocol to internal Service Controller messages, that the Orchestration Engine understands. Each interworking module provides connectivity to a single application or network entity, through its native protocol. For example, IM-ASF-SIP is used to communicate with a SIP application server through SIP. You deploy modules as you need, depending on the solution and use cases that you need to implement.

There are two types of interworking modules:

- Network-facing modules, that connect Service Controller to session control entities, such as Mobile Switching Centers (MSCs) and Call Session Control Functions (CSCFs). Network-facing modules provide a standard, stateful frontend, and act like applications toward session control entities. Session control entities communicate with Service Controller in the same way they would communicate with applications. For example, IM-SCF provides a frontend that appears to MSCs as a standard Service Control Point (SCP).
- Application-facing modules, that connect Service Controller to applications, such as IN SCPs and SIP application servers. Application-facing modules provide a standard, stateful frontend, and act like session control entities towards applications. Applications communication with Service Controller in the same way they would communicate with session control entities. For example, IM-ASF provides a frontend that appears to SIP applications as a standard CSCF.

Every module communicates on one end through a standard protocol with external session control entity or application, and on the other with the internal Orchestration Engine through the internal, proprietary, Session Abstraction Layer (SAL) protocol.

This section describes the types of modules available in Service Controller. For a detailed description of the modules, see *Service Controller Modules Configuration Guide*.

IM-SSF

IM-SSF is an application-facing module that enables Service Controller to connect to SCPs through an IN interface. You deploy IM-SSF to let an SCP control session in the legacy SS7-based network or IMS/LTE network. With IM-SSF, Service Controller acts like a standard Service Switching Point (SSP) towards the SCP, implementing the Service Switching Function (SSF) and generating IN triggers.

IM-SSF converts IN message to internal Service Controller messages, and the other way around. In addition to the protocol conversion functionality, IM-SSF supports:

- Initial and full session control modes

  SCPs control sessions in the SS7-based network or IMS network in one of the following modes:

  – Initial session control mode: SCPs control sessions through session setup only. IM-SSF generates IN triggers during session setup, and let the SCP instruct how to handle the session; that is continue or disconnect the session. IM-SSF does not request the session control entity to arm additional Detection Points (DPs), and simply stops controlling the session once the session setup has been completed.

  – Full session control mode: SCPs control sessions through session setup and then continue controlling the sessions until they terminate. After the session setup is complete, the IM-SSF instructs the session control entity to arm more DPs. IM-SSF continues controlling the session until it terminates.

- Originating and terminating BCSM

  IM-SSF fully implements the standard Basic Call State Model (BCSM) for both originating and terminating sessions.

- Media resources

IM-SCF supports both internal switch-based media resources (internal SRF) and external Intelligent Peripherals (IP) in the SS7 network. IM-SSF also supports MRFs in the IMS network. SCPs use media operations such as ConnectToResource(CTR) and EstablishTemporaryConnection(ETC) to play announcements and for user interactions.

- GGSN Data triggers

  IM-SSF fully supports GPRS control operations.

- Originating-side SMS triggers

  IM-SSF fully supports originating side SMS control operations.

- Configurative IN message and IN parameters tunnelling

  IM-SSF supports tunneling of IN information. In this mode the IN-SSF encodes IN messages and parameters using XML Encoding Rules (XER) or Binary Encoding Rules (BER), and delivers the encoded message to the IM-SCF inside the internal Service Controller SAL message.

- Switch-based charging timers and CDRs

  IM-SSF implements all SSF charging related timers. This capability enables an SCP to instruct IM-SSF to monitor call duration for online charging services, including prepaid services, and to insert charging information generated by IM-SSF into CDRs.

  You can configure the IM-SSF to perform the charging procedures. For example, to monitor call duration. Alternatively, IM-SSF can be coupled with the IM-SCF. In this case, IM-SSF instructs the IM-SCF to perform charging procedures.

- SCP management procedures

  IM-SSF supports operations such as ActivityTest, which allows SCPs to manage the availability of services to the network.

IM-SSF is available for a variety of protocols and protocol variants, including INAP, AIN, CAP and WIN.

IM-SCF (Reverse IM-SSF)

IM-SCF is a network-facing module that enables Service Controller to connect to Service Switching Points (SSPs) and Intelligent Peripherals (IPs) through an IN interface. You deploy IM-SCF to let applications control and initiate sessions in the legacy SS7-based network. With IM-SCF, Service Controller acts like a standard SCP towards the SSP, implementing the standard Service Control Function (SCF).

IM-SCF converts IN messages to internal Service Controller messages, and the other way around. In addition to the protocol conversion functionality, IM-SCF supports:

- Initial and full session control modes

  Applications control sessions in the SS7-based network in one of the following modes:

  – Initial session control mode: Applications control sessions through session setup only. When IM-SCF receives IN triggers for a session setup, Service Controller invokes applications and let them instruct how to handle the session; that is continue or disconnect the session. IM-SCF does not request the session control entity to arm additional Detection Points (DPs), and simply stops controlling the session once the session setup has been completed.

  - Full session control mode: Applications control sessions through session setup and then continue controlling sessions until they terminate. After the session setup is complete, the IM-SCF instructs the session control entity to arm more DPs. IM-SCF continues controlling the session until it terminates.

- Originating and terminating BCSM

  IM-SCF fully implements the standard Basic Call State Model (BCSM) for both originating and terminating sessions.

- Media resources

  IM-SCF supports both internal switch-based media resources (internal SRF) and external Intelligent Peripherals (IP). Applications can use resources in the SS7 network to play announcements and for user interactions. For example, application can collect subscriber input.

- Configurative IN message and IN parameters tunnelling

  IM-SCF supports tunneling of IN information. In this mode, the application is "IN-aware". That is the application encodes IN messages and parameters using XML Encoding Rules (XER) or Binary Encoding Rules (BER), and delivers the encoded message to the IM-SCF inside the body of the SIP message.

- Switch-based charging timers and CDRs

  Applications can control charging information in the session control entities, and leverage switch-based timers to implement online charging services.

- SCP management procedures

  IM-SCF supports operations such as ActivityTest, which allows applications to manage the availability of the application services to the network.

- Charging services

  IM-SCF can monitor session duration, generate credit reservation requests from application, and apply credit reservation responses on sessions.:

- SIP back-to-back, SIP proxy, and SIP redirect application servers

  IM-SCF can be configured to work with SIP applications in any mode: back-to-back, proxy, or redirect

IM-SCF is available for a variety of protocols and protocols variants, including INAP, AIN, CAP and WIN.

IM-ASF

IM-ASF-SIP is an application-facing module that enables Service Controller to connect to SIP applications through a SIP interface. You deploy IM-ASF to let SIP applications access the network and control sessions in the network.

Every instance of the IM-ASF module can communicates with only one application server.

R-IM-ASF

R-IM-ASF-SIP is a network-facing module that enables Service Controller to connect Call Session Control Functions (CSCFs) through a SIP interface. You deploy R-IM-ASF when you want your solution to control sessions in the IMS/LTE network, and invoke applications for those sessions. In addition to the basic SIP mediation functionality, R-IM-ASF supports:

- Charging service: in the absence of Diameter-based credit control application in the network, R-IM-ASF can monitor sessions, generate charging requests, analyze charging responses, and grant sessions with an approved time duration. R-IM-ASF can either monitor session duration, or delegate the monitoring to the CSCF.
- Media resources: a map of media resources in the network that application can use to play pre-call, mid-call, and post-call announcements.

IM-PSX ETSI/ANSI MAP

IM-PSX is a network-facing module providing Service Controller a MAP interface to HLRs and VLRs. You deploy IM-PSX if you want your solution to connect an HLR or a VLR, obtain information on subscriber's status and location, and let SIP applications use this information.

With IM-PSX SIP applications can:

- Query legacy SS7-based networks for information about subscribers, such as state, location, and the services the subscriber owns
- Modify subscriber information in the legacy SS7-based network (in GSM networks only)

IM-PSX is available for ETSI MAP and ANSI-41.

IM-UIX-SMS

IM-UIX-SMS is a network-facing module providing Service Controller a Short Message Peer-to-Peer Protocol (SMPP) interface to Short Message Service Centers (SMSCs). You deploy IM-UIX-SMS if you want your solution to communicate with an SMSC, and allow SIP applications send short messages to, and receive short messages from, subscribers in your network.

With IM-UIX-SMS, Service Controller acts as an External Short Message Entity (ESME), using the submit_sm and deliver_sm to communicate with the SMSC.

IM-UIX-SMS supports:

- Transmission of short messages to a single or multiple mobile subscribers
- Reception of short messages from mobile subscribers
- Scheduling delivery date and time of short messages
- Datagram and store-and-forward delivery modes
- Setting priorities for short message delivery
- Different types short message data coding
- Setting expiration time to short messages
- Direction of short messages at specific services

IM-UIX-SMS is available for SMPP 3.4.

IM-UIX-USSD

IM-UIX-USSD is a network-facing module providing Service Controller an SMPP interface to SMSCs for Unstructured Supplementary Services Data (USSD) delivery. You deploy IM-UIX-USSD if you want your solution to communicate with an SMSC, to allow SIP applications send USSD commands to subscribers in your network.

IM-UIX-USSD is available for SMPP 3.4.

IM-PSX-Plugin

IM-PSX-Plugin is a network-facing module that provides Service Controller with a TCAP interface to entities in the legacy SS7-based network. You deploy IM-PSX-Plugin if you want applications in your solution to communicate with network entities through protocols and using messages, which other modules such as IM-SSF, IM-SCF, and IM-PSX do not support.

IM-PSX-Plugin is available for ETSI and ANSI protocols.

Supplementary Modules

Service Controller includes the following supplementary modules:

Supplementary modules are interchangeable modules that facilitate and complement Service Controller solutions in certain deployments. The use of supplementary modules is optional.

SM-LSS

Local Subscriber Server (LSS) is an implementation of a subscriber profile server that can be used as a source for service orchestration logic. LSS stores subscriber profiles, including the orchestration logic that the Orchestration Engine uses to orchestrate applications for the subscriber's sessions. SM-LSS is available out of the box, deployed, and ready for use.

See *Service Controller Modules Configuration Guide* for detailed description of the SM-LSS.

SM-PME

SM-PME can be added to the orchestration chain, as if it was another application in the chain, to modify the content of messages to fit the requirements of the next application in the chain. For example, an application may require that the **Subject** header contains a particular value.

SM-PME can modify the headers and body of messages. You configure how SM-PME modifies the headers and body using an XML file that defines how SM-PME transforms the headers and body.

See *Service Controller Modules Configuration Guide* for detailed description of SM-PME.

About Implementing SIP Applications for the NGIN Solution

When deploying the Service Controller NGIN solution you can implement SIP applications that control and deliver services to sessions running in the legacy SS7-based network. Applications can redirect or disconnect a session, modify session information, implement charging services for sessions, play announcement to parties and do much more. For detailed description of the SIP interface provided by Service Controller to control session in the GSM network, see *Service Controller SIP Developer's Guide for GSM*.

You can implement SIP applications on any SIP application server, such as Oracle Communications Converged Application Server (OCCAS).

2

Setting Up the NGIN Solution

This chapter describes the Oracle Communications Service Controller Next Generation Intelligent Network (NGIN) solution and how to configure Service Controller for this solution.

About the NGIN Solution

You use the NGIN solution to deliver SIP services to sessions in the legacy SS7 network. Service Controller provides SIP applications with connectivity and access to the legacy SS7 network. Applications can control existing sessions and initiate new sessions in the network, query the state and location of mobile subscribers, and send SMS to subscribers.

SIP applications use the Service Controller's SIP interface to access and control sessions in the SS7 network. See *Service Controller SIP Developer's Guide for GSM* for detailed guidelines on developing SIP applications in the Service Controller NGIN solution.

Session Control

SIP applications use the Service Controller's SIP interface to get notified of new sessions. Acting as a back-to-back user agent, the SIP application operates between the two end points of the session, devices the signaling channel into two legs, and mediates all signaling between the two ends, from call establishment to termination. In this position, the SIP application controls the two legs of a session, and manages the session.

Figure 2–1 shows the basic session control call flow.

Figure 2–1 Basic Session Control Call Flow in the NGIN Solution

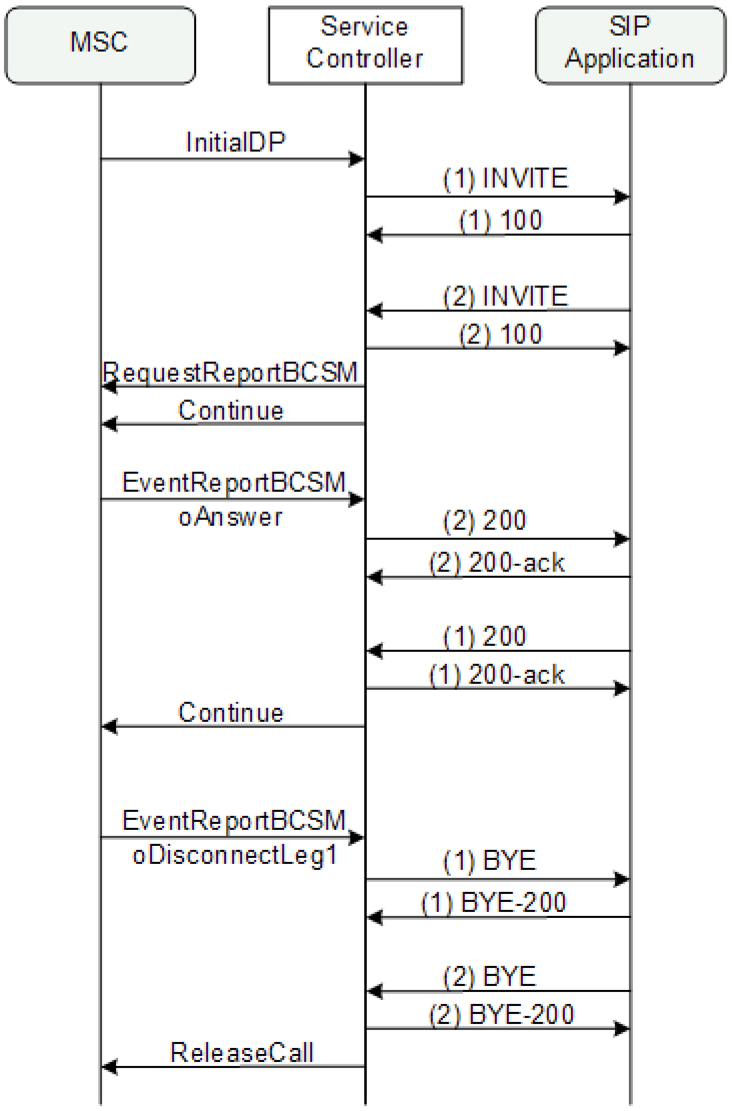
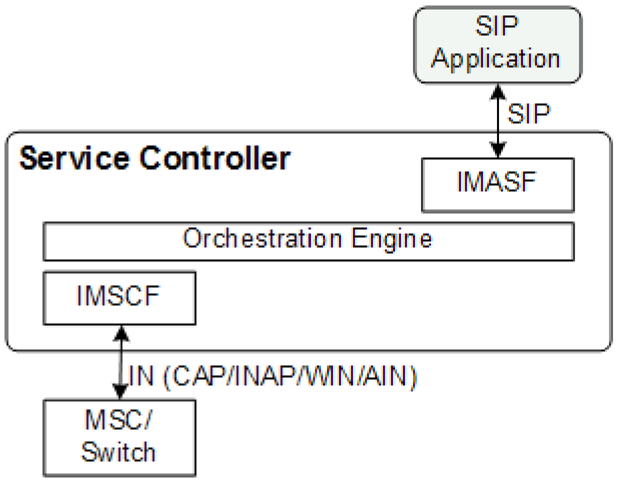


Figure 2–2 shows the components in the Service Controller processing tier that you set up to enable session control; IMASF, IMSCF, and the Orchestration Engine. You can use different variants of the IM-SCF module, depending on the protocol variant used in your network. For example, if your network uses CAP Phase 3, then you use the IM-SCF-CAP-Phase 3 module. To enable Service Controller's SIP connectivity to SIP applications and SS7 connectivity to entities in the SS7 network, you also need to configure the SIP SSU and SS7 SSU in the signaling tier.

Figure 2–2 Components Required for Session Control

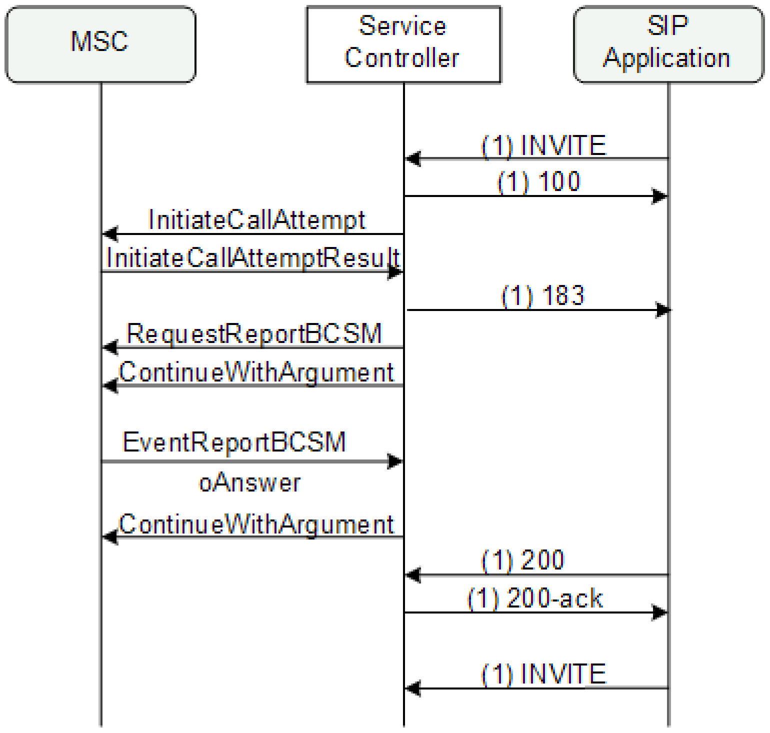


Session Initiation

In addition to controlling sessions initiated in the SS7 network, applications can initiate new session in the SS7 network. Figure 2–3 shows the basic session initiation call flow. In such call flows the SIP application acts as a SIP user agent.

To allow SIP application initiate sessions in the SS7 network, you set up the same components required for session control; IMASF, IMSCF, the Orchestration Engine, the SIP SSU, and the SS7 SSU. However, you need to make additional configurations to enable routing of incoming initial SIP messages into Service Controller.

Figure 2–3 Basic Session Initiation Call Flow in the NGIN Solution



Subscriber State and Location Query

SIP applications can use the Service Controller's SIP interface to query the state and location of mobile subscribers, obtain service subscription information from the HLR, and modify mobile subscriber's subscription information in the HLR and VLR.

Figure 2–4 show a basic state and location query call flow. In this call flow the SIP application uses SIP SUBSCRIBE and SIP NOTIFY messages, which Service Controller converts to MAP messages.

Figure 2–4 Basic Mobile Subscriber State Query Call Flow in the NGIN Solution

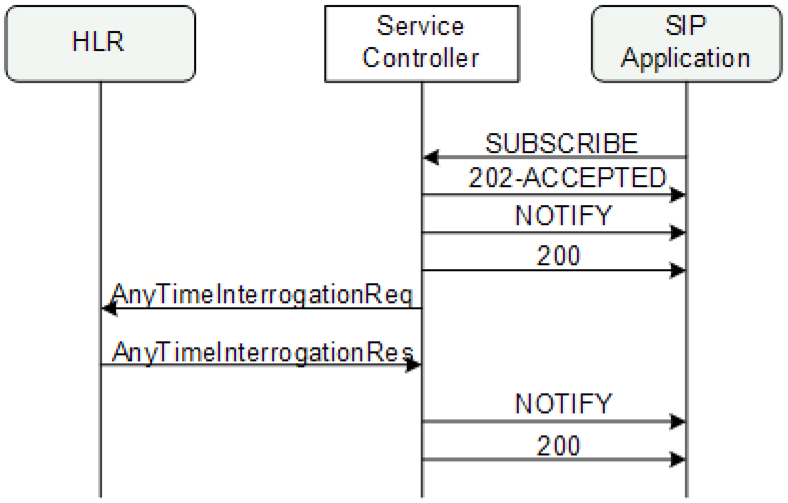


Figure 2–5 shows the components in the Service Controller processing tier that you set up to enable state and location query; IMASF, IMPSX, and the Orchestration Engine. You can use different variants of the IMPSX module, depending on the protocol variant used in your network. For example, for a GSM network you would use IMPSXMAP3. To enable Service Controller's SIP connectivity to SIP applications and SS7 connectivity the HLR and VLR in the SS7 network, you also need to configure the SIP SSU and SS7 SSU in the signaling tier.

If your system is already configured for the NGIN solution, and you extend the functionality of the system to also enable status and location query, you use the same IMASF module instance that is already deployed in the system, as shown in figure Figure 2–6.

Figure 2–5 Components Required for the State and Location Query

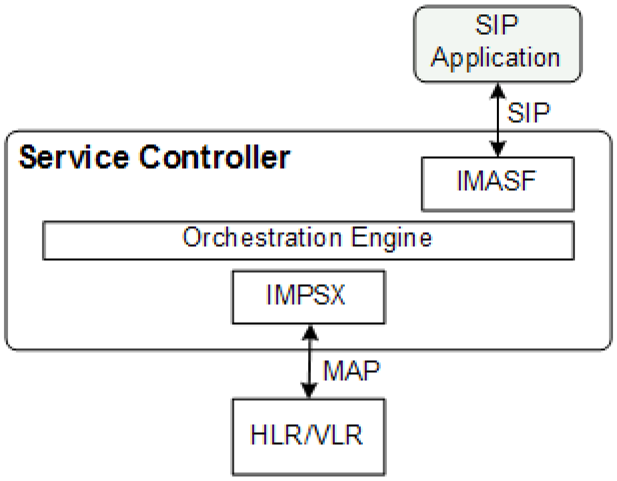
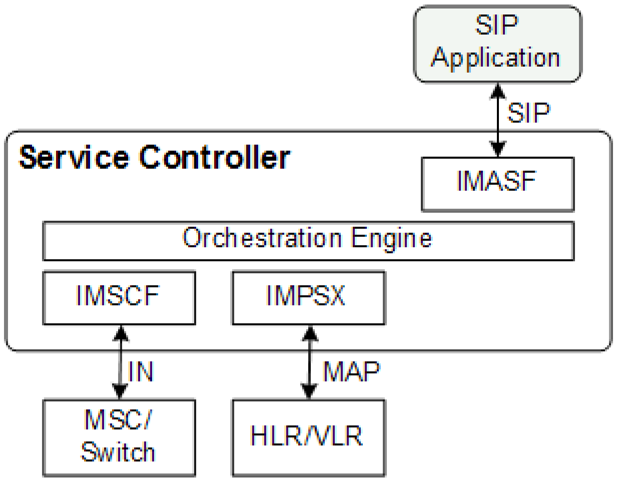


Figure 2–6 Components Required for the State and Location Query Combined with Session Control



Short Message Delivery

SIP applications can use the Service Controller's SIP interface to send short messages to, and receive short messages from mobile subscribers, through the mobile network's Short Message Service Center (SMSC). Figure 2–7 shows the basic call flow for delivering a short message from the application to the mobile subscriber. In this call flow, the SIP application uses SIP MESSAGE messages which Service Controller converts to SMPP submit_sm message and forwards to the SMSC. The SMPP deliver_sm message is optional and will only be generated if the solution is configured to request acknowledgement SMPP messages from the SMSC. Figure x-x shows the basic call flow for reception of s short message from a mobile subscriber.

Figure 2–7 Basic Short Message Delivery Call Flow in the NGIN Solution

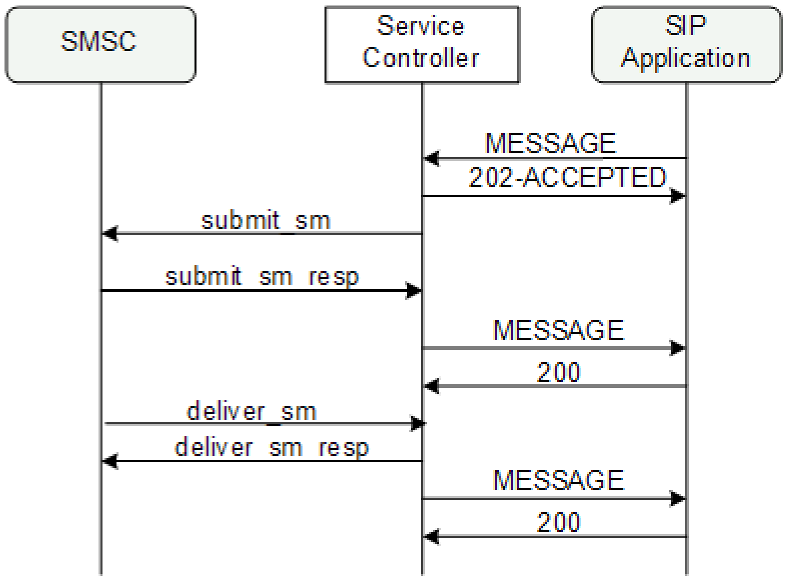


Figure 2–8 Basic Short Message Reception Call Flow in the NGIN Solution

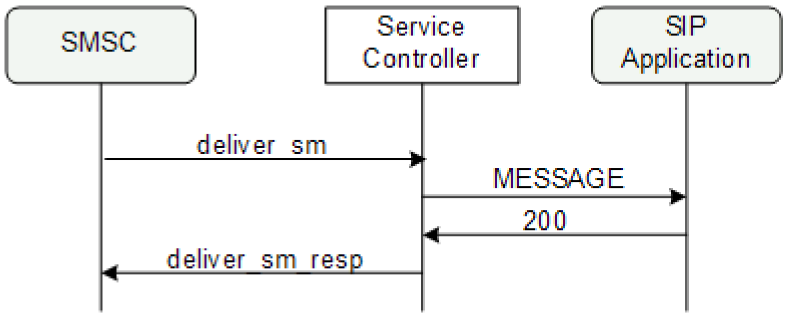
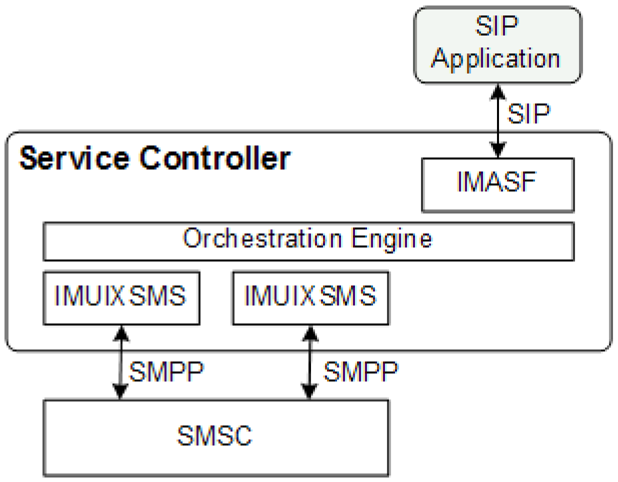


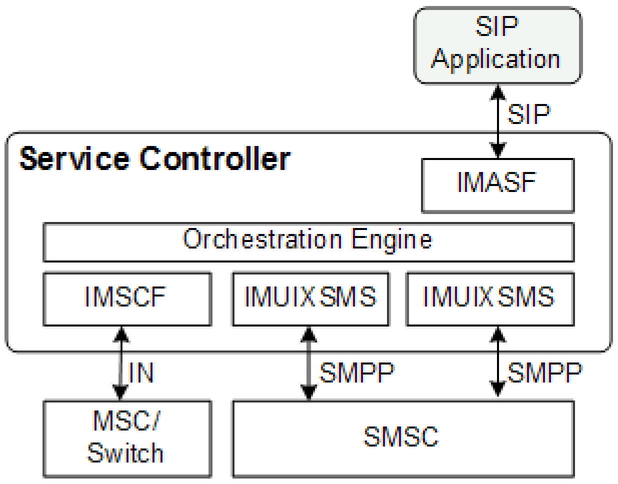
Figure 2–9 shows the components in the Service Controller processing tier that you set up to enable SMS delivery; IMASF, IMUIXSMS, and the Orchestration Engine. An instance of IMUIXSMS supports delivery of messages in one direction only; That is, either from the application to the network, or from the network to the application. Therefore, to enable message delivery in both direction simultaneously, deploy two instances of IMUIXSMS. In this use case, you also configure the SIP SSU and SMPP SSU in the signaling tier., to enable elementary stack-level connectivity with the SIP application and SMSC.

Figure 2–9 Components Required for SMS Delivery



If your system is already configured for the NGIN solution, and you extend the functionality of the system to also enable message delivery, you use the same IMASF module instance that is already deployed in the system, as shown in figure Figure 2–6.

Figure 2–10 Components Required for SMS Delivery Combined with Session Control



Enabling Control of Sessions in the SS7 Network

An end-to-end configuration that enables SIP applications control sessions in the SS7 network requires setting up IMASF, IMSCF, the Orchestration Engine, SS7 SSU, and SIP SSU.

To set up this configuration:

1. Enable Service Controller to accept traffic of sessions arriving from the legacy SS7 network. See "Connecting Service Controller to the Legacy SS7 Network" for information about the components that you need to configure and how to configure them.
2. Enable the Service Controller SIP interface to allow SIP applications control sessions that run through Service Controller. See "Setting Up the SIP Interface for SIP Applications" for more information.

3. Route sessions that arrive from the legacy SS7 network to the SIP applications. See "Defining a Service Orchestration Chain" for more information.

Connecting Service Controller to the Legacy SS7 Network

You can connect Service Controller to a SIGTRAN-based SS7 network. Service Controller does not support TDM-based SS7 networks.

Before you start, make sure that you have a detailed plan of how Service Controller connects to your SS7 network, including point codes that you assign to Service Controller. To connect Service Controller to the legacy SS7 network:

1. Configure a SIGTRAN-based SS7 SSU for your SIGTRAN-based SS7 network. See *Service Controller Signaling Server Units Configuration Guide* for more information.
2. Deploy and configure the type of IM-SCF module that suits the specific IN protocol of your network, as described in *Service Controller Modules Configuration Guide*, in the relevant IMSCF chapter.
3. Configure routing rules in the SS7 SSU to route sessions arriving from the SS7 network to the IMSCF module that you deployed in step 2.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.
   b. Select **SSU SS7 SIGTRAN**.
   c. In the **Routing** tab, in the left pane, click the **Add** button. The **New** dialog box appears.
   d. In the **Name** field, enter a name for the routing rule. Click **Apply**. The newly created rule now appears in the rules tree in the left pane.
   e. Select the newly created rule node, and then select the **Incoming Routing Rules** tab.
   f. In the **Module Instance** field enter *module*.*type*@*domain* where *module* is the name of the IMSCF module that you deployed in step 2, *type* is the type of the IMSCF module that you deployed, and *domain* is the name of the domain where you deployed the module. For example, imscf.IMSCFCAP3@ocsb.
   g. Optionally, you can define criteria for the rule, so that the SSU SS7 route to the SIP application only sessions that meet the criteria. You define criteria for the rule in the **Incoming Routing Rule** Criteria tab.

Setting Up the SIP Interface for SIP Applications

To enable the Service Controller SIP interface:

1. Configure Service Controller as a SIP entity, as described in *Service Controller Signaling Server Units Configuration Guide*, in the chapter about the SIP SSU.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.
   b. Select the **SSU SIP** node.
   c. In the **SSU SIP** tab, select the **SIP Server** tab.
   d. In the **Globally Routable User Agent URI** field, enter a user agent identifier that uniquely represents Service Controller in the network. Enter sip:*ip*:*port*,

where *ip* is an alpha-numeric IP-address or DNS name, and *port* is the numeric port that Service Controller uses to listen to SIP responses.

2. (Optional) Configure the information of the SIP application server where your SIP application is running. In the SIP SSU, define the SIP application as a SIP network entity.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.
   b. Select the **SSU SIP** node.
   c. In the **SSU SIP** tab, select the **SIP Network Entities** tab.
   d. Click the **New** button. The **New** dialog box appears.
   e. Enter the fields in the New dialog box as described in the section about configuring SIP network entities in *Service Controller Signaling Server Units Configuration Guide*.
   f. Click **Apply**.

3. Deploy and configure the IMASF module as described in *Service Controller Modules Configuration Guide*, in the chapter about the IMASF module.

   Specifically, configure the address of the SIP application server where your applications are running:

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node.
   b. Expand the **Processing Tier** node, and then the **Interworking Modules** node.
   c. Select the node of the IMASF module that you deployed.
   d. In the **Configuration** tab, select the **Application Server** tab.
   e. In the **AS Address Alias** field enter sip:*ip*:*port*, where *ip* is an alpha-numeric IP-address or DNS name of the application server, and *port* is the numeric port that the application server listens to.

      Alternatively, you can enter sip:*alias*, where *alias* is the alias of a SIP network entity that you configured previously in the SIP SSU.

Defining a Service Orchestration Chain

To route sessions to the SIP application server, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain. Include the SIP application server in the orchestration chain. See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

Enabling Initiation of Sessions in the SS7 Network

To allow SIP applications initiate sessions, in addition to controlling sessions initiated in the SS7 network, you perform additional configuration beyond the configuration described in "Enabling Control of Sessions in the SS7 Network". The additional configuration includes:

1. Enabling Service Controller to accept initial SIP messages of sessions initiated by the SIP application. You do that by configuring SIP network access points in the

SIP SSU, in the signaling tier, through which new incoming SIP sessions arrive. See "Enabling Acceptance of Incoming Initial SIP Messages" for information about configuring SIP network access points.

2. Routing new incoming SIP sessions that arrive through the SIP SSU to the IMASF module. See "Routing Incoming Initial SIP Messages to the IMASF Module" for instructions. The IMASF module converts the SIP messages to internal Service Controller messages and pass them to the Orchestration Engine.
3. Routing the sessions that arrive to the Orchestration Engine, through the IMSCF module, to the legacy SS7 network. See "Routing Sessions to Switches in the SS7 Network" for more information.

Enabling Acceptance of Incoming Initial SIP Messages

To enable Service Controller accept incoming initial SIP messages, configure each server in the signaling tier as a SIP network access point.

In the Administration Console:

1. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
2. Select the **SSU SIP** node.
3. In the **SIP** tab, in the **SIP Configuration** tab, select the **Network Access Points** tab.
4. Click the **Add** button. The **New** dialog box appears.
5. Enter an alpha-numeric name for the newly added network access point. Click **Apply**. A new network access point appears in the tree.
6. Select the recently added network access point.
7. In the **General** tab, in the **Target** field, enter the name of the server and click **Apply**.
8. Select the **Listen Address** tab, configure the **Host** and **Port** of the network access point, and click **Apply**.
9. Select the **External Listen Address** tab. If you are using a **Load Balancer** in your system, configure the **Host** and **Port** of the Load Balancer. Otherwise, configure again the **Host** and **Port** of the network access point.

Repeat steps **4** through **9** for every server in the signaling tier.

For more details about configuring network access points in the SIP SSU, see the chapter about the SIP SSU in *Service Controller Signaling Server Units Configuration Guide*.

Routing Incoming Initial SIP Messages to the IMASF Module

To route incoming SIP messages from your SIP applications to the IMASF module, you configure incoming routing rules in the SIP SSU.

In the Administration Console:

1. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
2. Select the **SSU SIP** node.
3. In the **SSU SIP** tab, select the **Incoming Routing Rules** tab.
4. Click the **New** button. The **New** dialog box appears.
5. In the **Name** field, enter a name for the rule.

6. In the **IP Address** field, enter the address of your SIP application.
7. In the **Alias** field, enter the alias of the IMASF module that you want to process the messages from your SIP application. Enter ssu:*IMASF-name*.IMASF@*domain* where *IMASF-name* is the name that you gave to the IMASF module when you deployed it, and *domain* is the name of the processing domain where the module is deployed.

For more details about configuring incoming routing rules in the SIP SSU, see the chapter about the SIP SSU in *Service Controller Signaling Server Units Configuration Guide*.

Routing Sessions to Switches in the SS7 Network

To route sessions arriving to the Orchestration Engine, toward the SS7 network, you define an orchestration logic that route messages through the IMSCF module, to the legacy SS7 network. In the orchestration logic you apply criteria on arriving sessions, so that the Orchestration Engine route to the SS7 network only the sessions initiated by the SIP application. Use the Orchestration Studio to define an orchestration logic

If you are using the LSS as a source of orchestration logic, update the orchestration logic of the LSS orchestration profile that you are using. See "Updating the Orchestration Logic in the LSS" for more information.

If you are using the HSS as a source for orchestration logic, you need to use the tools provided with the HSS to change the orchestration logic. Alternatively, you can use the default orchestration profile in the Orchestration Studio to define the updated orchestration logic. Then, to provision the updated orchestration logic in the HSS, you use tools provided by the HSS.

Enabling Query of Subscriber State and Location

An end-to-end configuration that enables SIP applications query the state and location of mobile subscribers, requires setting up IMASF, IMPSX, the Orchestration Engine, SS7 SSU, and SIP SSU.

To set up this configuration:

1. Enable Service Controller to access the HLR and VLRs in the mobile SS7 network. See "Connecting Service Controller to the Legacy SS7 Network" for information about the components that you need to configure and how to configure them.
2. Enable the Service Controller SIP interface to allow SIP applications query the state and location of mobile subscribers. See "Setting Up the SIP Interface for SIP Applications" for more information.
3. Route requests from the SIP application to the mobile network. See "Routing SIP Requests to the HLR and VLR in the Mobile Network" for more information.

Connecting Service Controller to the Mobile Network

You can connect Service Controller to a SIGTRAN-based SS7 network. Service Controller does not support TDM-based SS7 networks.

Before you start, make sure that you have a detailed plan of how Service Controller connects to your SS7 network, including point codes that you assign to Service Controller. To connect Service Controller to the legacy SS7 network:

1. Configure a SIGTRAN-based SS7 SSU for your SIGTRAN-based SS7 network. See *Service Controller Signaling Server Units Configuration Guide* for more information.

Note: For state and location query, where sessions are always initiated by the SIP application, you do not need to configure the SS7 SSU to route new incoming SS7 sessions. Therefore, there is no need to configure the **Routing** tab in the SS7 SSU configuration screens.

---

2. Deploy and configure the type of IMPSX module that suits the particular protocol of your network; For an ETSI network deploy IMPSXMAP3, and for an ANSI network deploy IMPSXANSIMAP. For detailed information about deploying and configuring IMPSX, see the chapter about the relevant IMPSX module in *Service Controller Modules Configuration Guide*.

Setting Up the SIP Interface for SIP Applications

To enable the Service Controller SIP interface:

1. Deploy and configure the IMASF module as described in *Service Controller Modules Configuration Guide*, in the chapter about the IMASF module.
2. Enable Service Controller to accept initial SIP messages of sessions initiated by the SIP application. You do that by configuring SIP network access points in the SIP SSU, in the signaling tier. See "Enabling Acceptance of Incoming Initial SIP Messages" for information about configuring SIP network access points.
3. Route new incoming SIP sessions that arrive through the SIP SSU, to the IMASF module. See "Routing Incoming Initial SIP Messages to the IMASF Module" for instructions. The IMASF module converts the SIP messages to internal Service Controller messages and passes them to the Orchestration Engine.

Routing SIP Requests to the HLR and VLR in the Mobile Network

To route requests in the Orchestration Engine to an HLR or VLR in the SS7 network, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain.

Route messages through the IMPSX module, toward the legacy SS7 network. In the orchestration logic you apply criteria on arriving requests, so that only state and location queries, that is SUBSCRIBE requests, be routed through IMPSX.

See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

Enabling Short Message Delivery to Mobile Subscribers

An end-to-end configuration that enables SIP applications send short message to mobile subscribers, requires setting up IMASF, IMUIXSMS, the Orchestration Engine, SMPP SSU, and SIP SSU.

To set up this configuration:

1. Enable the Service Controller SIP interface to allow SIP applications send short messages toward the network. See "Setting Up the SIP Interface for SIP Applications" for more information.
2. Set up the Service Controller SMPP interface to enable conversion of short messages to SMPP, and delivery of short messages to the SMSC. See "Connecting Service Controller to the SMSC" for more information.

3. Route short messages that arrive from the SIP application to the SMSC. See "Routing SIP Requests to the SMSC" for more information.

Setting Up the SIP Interface for SIP Applications

To enable the Service Controller SIP interface:

1. Deploy and configure the IMASF module as described in *Service Controller Modules Configuration Guide*, in the chapter about the IMASF module.
2. Enable Service Controller to accept initial SIP messages of sessions initiated by the SIP application. You do that by configuring SIP network access points in the SIP SSU, in the signaling tier. See "Enabling Acceptance of Incoming Initial SIP Messages" for information about configuring SIP network access points.
3. Route new incoming SIP sessions that arrive through the SIP SSU, to the IMASF module. See "Routing Incoming Initial SIP Messages to the IMASF Module" for instructions. The IMASF module converts the SIP messages to internal Service Controller messages and passes them to the Orchestration Engine.

Connecting Service Controller to the SMSC

To connect Service Controller to the SMSC you configure the SMPP SSU and IMUIXSMS module.

In the SMPP SSU in the signaling tier, configure the connection of Service Controller with the SMSC. In this connection, Service Controller acts as an External Short Message Entity (ESME). For detailed information on how to configure the SMPP SSU see the chapter about the SMPP SSU, in *Service Controller Signaling Server Units Configuration Guide*.

Specifically for message delivery, the minimum configuration that you perform in the SMPP SSU includes:

1. Defining the credentials required to secure the connectivity with the SMSC:
   a. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
   b. Select the **SSU SMPP** node.
   c. In the **SMPP** tab, select the **Credential Store** tab.
   d. In the **Password** area, in the **Password** field, enter the password that you configured in the SMSC for the connection with Service Controller.
   e. In the **Password** area, in the **Key** field, enter a numeric identifier for the connection password.
   f. In the **Password** area, clear the **One-way** checkbox.
   g. Click the **Set** button. The password is stored in the Credential Store, and can be assigned to connections with SMSC, when you define the connections, by referencing the **Key** field of the password.
2. Defining the address of the SMSC and the SMSC connection parameters:
   a. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
   b. Select the **SSU SMPP** node.
   c. In the **SMPP** tab, select the **SMSC** tab.
   d. Click the **New** button. The **New** dialog box appears.

e. In the **SMSC Identifier** field, enter a unique identifier for the SMSC that you want to connect.

f. In the **SMSC Address** and **SMSC Port** fields, enter the IP address and port of the SMSC.

g. In the **ESME Credential Key** field, enter the key that you assigned for the SMSC password in the Credential Store. Enter the numeric identifier that you assigned for the SMSC password in step e.

h. Click **Apply**. The connection with the SMSC now appears in the SMSC table.

Repeat steps **a** through **h** for every SMSC instance that you want Service Controller to connect.

3. Configuring the SMSCs whose connections you defined in step 2 as a valid destination that Service Controller maintains an open connection with:

   a. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.

   b. Select the **SSU SMPP** node.

   c. In the **SSU SMPP** tab, select the **SMPP Network Entities** tab.

   d. Click the **New** button. The **New** dialog box appears.

   e. In the **Name** field, enter a unique alpha-numeric identifier for the SMSC.

   f. In the **Alias** field, enter an alias that you to assign to the destination SMSC. You can enter the same alias later when you define more destination SMSCs, to have a number of destination SMSCs share load.

   g. In the **SMSC** field, enter the ID of an SMSC instance that you defined in step 2.

   h. Click the **Apply** button. The new SMSC destination now appears in the SMPP Network Entities table.

In the processing tier, deploy and configure an IMUIXSMS module as discussed in the chapter about IMUIXSMS in *Service Controller Modules Configuration Guide*.

In the IMUIXSMS module, the minimum configuration that you perform includes:

1. Specifying the format of short messages that arrive from the SIP application:

   a. In the navigation tree, expand the **OCSB** node, then the **Processing Tier** node, and then the **Interworking Modules** node.

   b. Select the instance of the IMUIXSMS module that you deployed.

   c. In the **Configuration tab** select the **General** tab.

   d. From the **Body Encoding Format** list, select the format of short messages when they arrive to Service Controller, from the SIP application, in the body of the SIP messages. If short messages are encoded in either XER or BER, and are ready to be forwarded as is to the network, select XER or BER respectively. If messages are not encoded, that is they appear as plain text, select NONE.

   e. Click the **Apply** button.

2. Specifying the destination SMSC for messages sent by the SIP application:

   a. In the navigation tree, expand the **OCSB** node, then the **Processing Tier** node, and then the **Interworking Modules** node.

   b. Select the instance of the IMUIXSMS module that you deployed.

c. In the **Configuration tab** select the **SMPP Handling** tab. In this tab you configure how Service Controller construct submit_sm messages.
d. In the **Service Type** field, enter the identifier of the destination SMS service.
e. In the **SMSC Alias** field, enter the alias of an SMSC that you have previously defined in the SMPP SSU, in step 3.
f. Click the **Apply** button.

Routing SIP Requests to the SMSC

To route requests in the Orchestration Engine to the SMSC in the network, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain.

Define an orchestration logic that route messages through the IMUIXSMS module, toward the network. In the orchestration logic you apply criteria on arriving requests, so that only short message delivery requests, that is MESSAGE requests, be routed through IMUIXSMS.

See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

Enabling Short Message Reception from Mobile Subscribers

An end-to-end configuration that enables SIP applications receive short message from mobile subscribers, requires setting up IMASF, IMUIXSMS, the Orchestration Engine, SMPP SSU, and SIP SSU.

To set up this configuration:

1. Enable Service Controller to accept SMSs arriving from the SMSC. Set up the Service Controller SMPP interface to enable reception of SMSs, and conversion of Sums to SIP short messages. See "Connecting Service Controller to the SMSC" for more information.
2. Enable the Service Controller SIP interface to allow SIP applications receive short messages that arrive from the network. See "Setting Up the SIP Interface for SIP Applications" for more information.
3. Route SMSs that arrive from the network towards the SIP application. See "Routing SMSs to the SIP Application" for more information.

Connecting Service Controller to the SMSC

To connect Service Controller to the SMSC follow the same instructions for "Connecting Service Controller to the SMSC" described in the use case of short message delivery.

Additionaly, configure routing rules in the SMPP SSU to route SMSs that arrive from the SMSC to the IMUIXSMS module. Define the criteria for the rule, so that the SMPP SSU route only SMSs that meet the criteria.

In the Administration Console:

1. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
2. Select the **SSU SMPP** node.
3. In the **SSU SMPP** tab, select the **Incoming Routing Rules** tab.

4. Click the **New** button. The **New** dialog box appears.
5. In the **Name** field, enter a name for the rule.
6. In the **SMPP Destination Address** and **Service Type** fields, enter the criteria for the rule. That is the SMS destination address, and the SMS destination service.
7. In the **Alias** field, enter the alias of the IMUIXSMS module that you want to process the SMSs. Enter ssu:*IMUIXSMS-name*.IMUIXSMSSMPP34@*domain* where *IMUIXSMS-name* is the name that you gave to the IMUIXSMS module when you deployed it, and *domain* is the name of the processing domain where the module is deployed.

For more details about configuring incoming routing rules in the SMPP SSU, see the chapter about the SMPP SSU in *Service Controller Signaling Server Units Configuration Guide*.

Note: To set up a system that support short message delivery and short message reception simultaneously, you must use a different instance of the IMUIXSMS module for each use case.

Setting Up the SIP Interface for SIP Applications

Follow the instructions in "Setting Up the SIP Interface for SIP Applications" to enable the Service Controller SIP interface.

Routing SMSs to the SIP Application

To route requests in the Orchestration Engine to the SIP application, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain.

Define an orchestration logic that route messages through the IMASF module, to the SIP application. In the orchestration logic you apply criteria on arriving requests, so that only short message delivery requests, that is MESSAGE requests, be routed through IMASF.

Note: To set up a system that support short message delivery and short message reception simultaneously, you must define an orchestration logic that route MESSAGE requests differently, depending on the origin of the request. Route MESSAGE requests that arrive from the SIP application, toward the network. Route MESSAGE requests that arrive from the SMSC, to the SIP application. Use a request’s origin, for example, the SIP From header, as the base criteria for routing.

See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

3

Setting Up the SCIM Solution

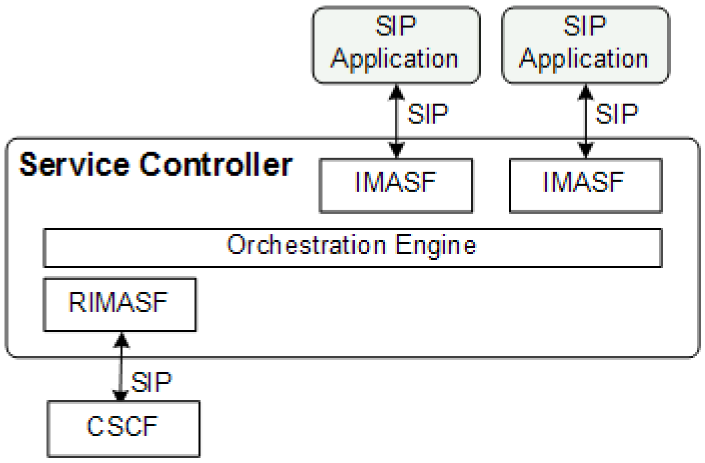
This chapter describes the Oracle Communications Service Controller Service Capability Interaction Manager (SCIM) solution and how to configure Service Controller for this solution.

About the SCIM Solution

You use the SCIM solution to combine the service logic of multiple applications, and deliver the combined services to sessions in the network. Service Controller supports mixed orchestration of SIP-based and IN-based applications. The use case in this chapter demonstrates orchestration of multiple SIP-based services, and how to apply them to sessions in the IMS network. However, you can also combine IN-based services with SIP-based services and apply them to sessions in the legacy SS7 network.

Figure 3–1 shows the Service Controller components in the processing tier that you set up in order to combine two SIP-based services and apply them to sessions in the IMS network: two instances of IMASF, one to connect each application, RIMASF, and the Orchestration Engine. You also need to configure the SIP SSU in the signaling tier to enable SIP connectivity with the CSCF and SIP applications.

Figure 3–1 The Service Controller Components Required for the SCIM Solution



When combining the service logic of two applications, Service Controller routes sessions through the two applications, sequentially. Each application acts as back-to-back SIP user agent. Requests from the calling party pass through one application, then through the second application, and then directed to the called party, as shown in Figure 3–2. Responses from the called party pass through the second application, then through the first application, and then reach the calling party.

Figure 3–2 Applications Orchestration Order

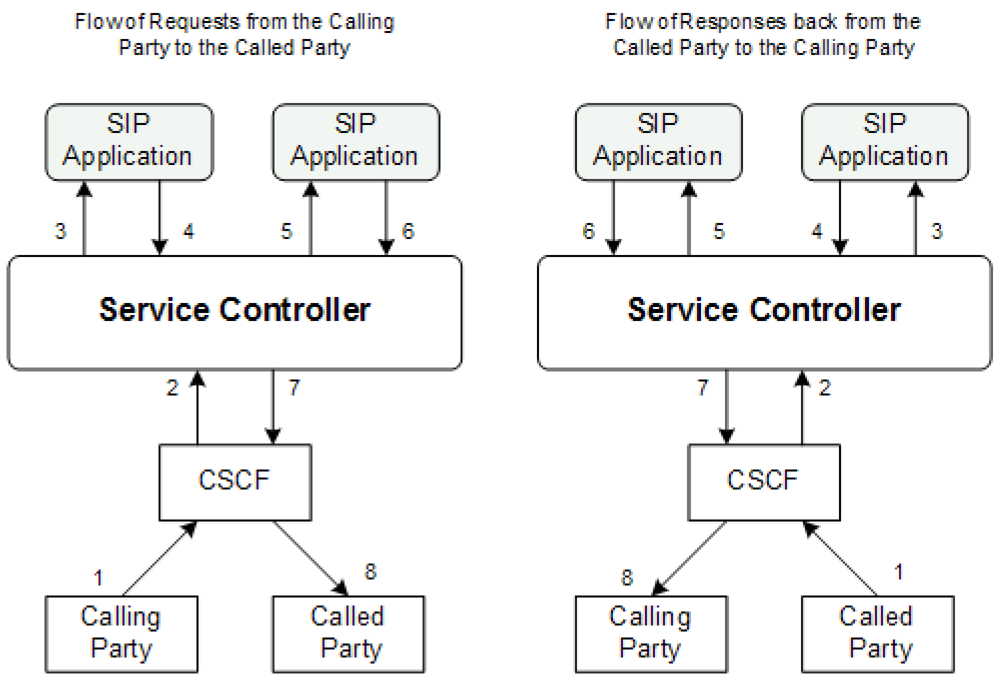
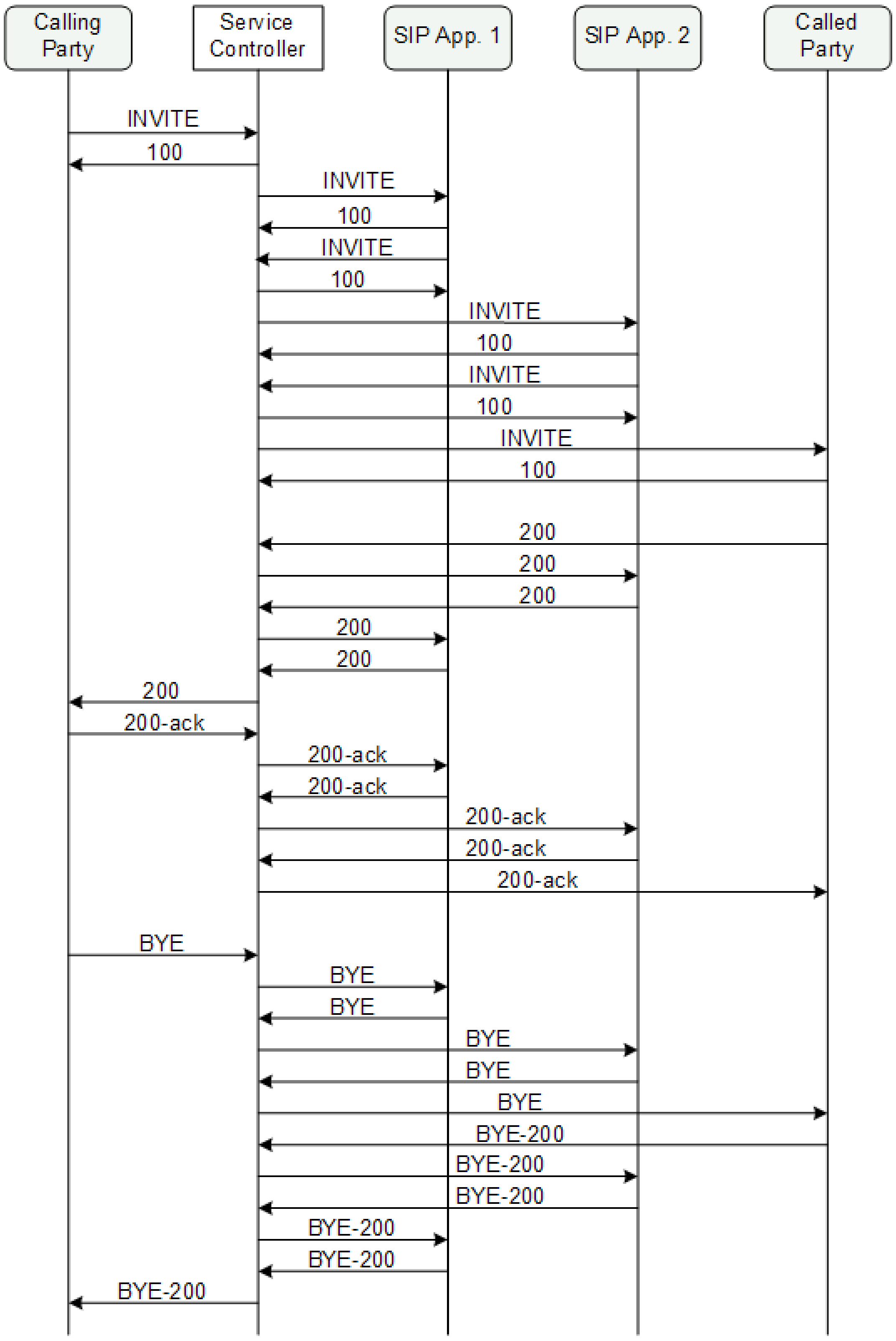


Figure 3–3 shows the full basic call flow for the SCIM use case shown in Figure 3–1.

Figure 3–3 Basic SCIM Solution Call Flow



Enabling Delivery of Multiple Applications to Sessions in the Network

This section describes how to set up an end-to-end configuration that combines two applications and deliver them to sessions in the network. For realization, this section walks you through the configuration of the solution in Figure 3–1; Orchestration of two SIP applications and delivery of the combined service logic to sessions in the IMS network. The documentation indicates steps that you can alter to implement a different solutions.

To set up an end-to-end configuration that delivers combined SIP applications logic to sessions in the IMS:

1. Enable Service Controller to accept traffic of sessions arriving from the IMS network. See "Connecting Service Controller to the IMS Network" for information about the components that you need to configure and how to configure them.

   To deliver combined service logic to sessions in the SS7 network, connect Service Controller to the SS7 network instead. See "Connecting Service Controller to the Legacy SS7 Network" for information.

2. Enable the Service Controller SIP interface with one SIP application, and then with the second SIP application. For each SIP application, repeat the instructions in "Setting Up the SIP Interface with SIP Applications" once.

3. Route sessions that arrive from the network through the applications. See "Defining a Service Orchestration Chain" for information.

Connecting Service Controller to the IMS Network

To connect Service Controller to the IMS network:

1. Configure Service Controller as a SIP entity, as described in *Service Controller Signaling Server Units Configuration Guide*, in the chapter about the SIP SSU.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.

   b. Select the **SSU SIP** node.

   c. In the **SSU SIP** tab, select the **SIP Server** tab.

   d. In the **Globally Routable User Agent URI** field, enter a user agent identifier that uniquely represents Service Controller in the network. Enter sip:*ip*:*port*, where *ip* is an alpha-numeric IP-address or DNS name, and *port* is the numeric port that Service Controller uses to listen to SIP responses.

2. Configure each server in the signaling tier as a SIP network access point.

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.

   b. Select the **SSU SIP** node.

   c. In the **SIP** tab, in the **SIP Configuration** tab, select the **Network Access Points** tab.

   d. Click the **Add** button. The **New** dialog box appears.

   e. Enter an alpha-numeric name for the newly added network access point. Click **Apply**. A new network access point appears in the tree.

   f. Select the recently added network access point.

g. In the **General** tab, in the **Target** field, enter the name of the server and click **Apply**.

h. Select the **Listen Address** tab, configure the **Host** and **Port** of the network access point, and click **Apply**.

i. Select the **External Listen Address** tab. If you are using a **Load Balancer** in your system, configure the **Host** and **Port** of the Load Balancer. Otherwise, configure again the **Host** and **Port** of the network access point.

Repeat steps **d** through **i** for every server in the signaling tier.

For more details about configuring network access points in the SIP SSU, see the chapter about the SIP SSU in *Service Controller Signaling Server Units Configuration Guide*.

3. Route incoming SIP messages from the network to the R-IMASF module

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.

   b. Select the **SSU SIP** node.

   c. In the **SSU SIP** tab, select the **Incoming Routing Rules** tab.

   d. Click the **New** button. The **New** dialog box appears.

   e. In the **Name** field, enter a name for the rule.

   f. In the **IP Address** field, enter the address of your SIP application.

   g. In the **Alias** field, enter the alias of the IMASF module that you want to process the messages from your SIP application. Enter ssu:*IMASF-name*.IMASF@*domain* where *IMASF-name* is the name that you gave to the IMASF module when you deployed it, and *domain* is the name of the processing domain where the module is deployed.

   For more details about configuring incoming routing rules in the SIP SSU, see the chapter about the SIP SSU in *Service Controller Signaling Server Units Configuration Guide*.

4. Deploy and configure the R-IMASF module as described in *Service Controller Modules Configuration Guide*, in the chapter about R-IMASF.

Setting Up the SIP Interface with SIP Applications

To enable the Service Controller SIP interface:

1. (Optional) Configure the information of the SIP application server where your SIP application is running. In the SIP SSU, define the SIP application as a SIP network entity.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.

   b. Select the **SSU SIP** node.

   c. In the **SSU SIP** tab, select the **SIP Network Entities** tab.

   d. Click the **New** button. The New dialog box appears.

   e. Enter the fields in the **New** dialog box as described in the section about configuring SIP network entities in *Service Controller Signaling Server Units Configuration Guide*.

f. Click **Apply**.

2. Deploy and configure the IMASF module as described in *Service Controller Modules Configuration Guide*, in the chapter about the IMASF module.

   Specifically, configure the address of the SIP application server where your applications are running.

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node.

   b. Expand the **Processing Tier** node, and then the **Interworking Modules** node.

   c. Select the node of the IMASF module that you deployed.

   d. In the **Configuration** tab, select the **Application Server** tab.

   e. In the **AS Address Alias** field enter sip:*ip*:*port*, where *ip* is an alpha-numeric IP-address or DNS name of the application server, and *port* is the numeric port that the application server listens to.

      Alternatively, if you performed step 1 and already configured the address of the SIP application in the SIP SSU, enter the alias that you assigned to the SIP application. Enter sip:*alias*, where *alias* is the alias that you assigned to the SIP application.

Defining a Service Orchestration Chain

To route sessions through multiple applications, you have to first choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain with the multiple applications in it. See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

4 Setting Up Online Charging for SS7 Networks

This chapter describes the Oracle Communications Service Controller online charging for SS7 networks solution and how to configure Service for this solution.

About the Online Charging for SS7 Networks Solution

You use the online charging for SS7 networks solution to deliver Diameter-based charging for subscriber sessions in the legacy SS7 network. Service Controller delivers SS7 sessions to online charging systems (OCSs) including Oracle Communications Billing and Revenue Management (BRM) and third-party charging systems supporting the Diameter Ro protocol.

Service Controller supports online charging for SS7 networks with a connection to an OCS through the Diameter Ro protocol.

Figure 4–1 shows a basic session charging flow of the online charging for SS7 networks solution. Service Controller supports additional call flow scenarios not shown such as applying charging after answer and internal charging for CAP1, WIN and AIN networks.

Figure 4–1 Basic Call Flow in the Online Charging for SS7 Networks Solution

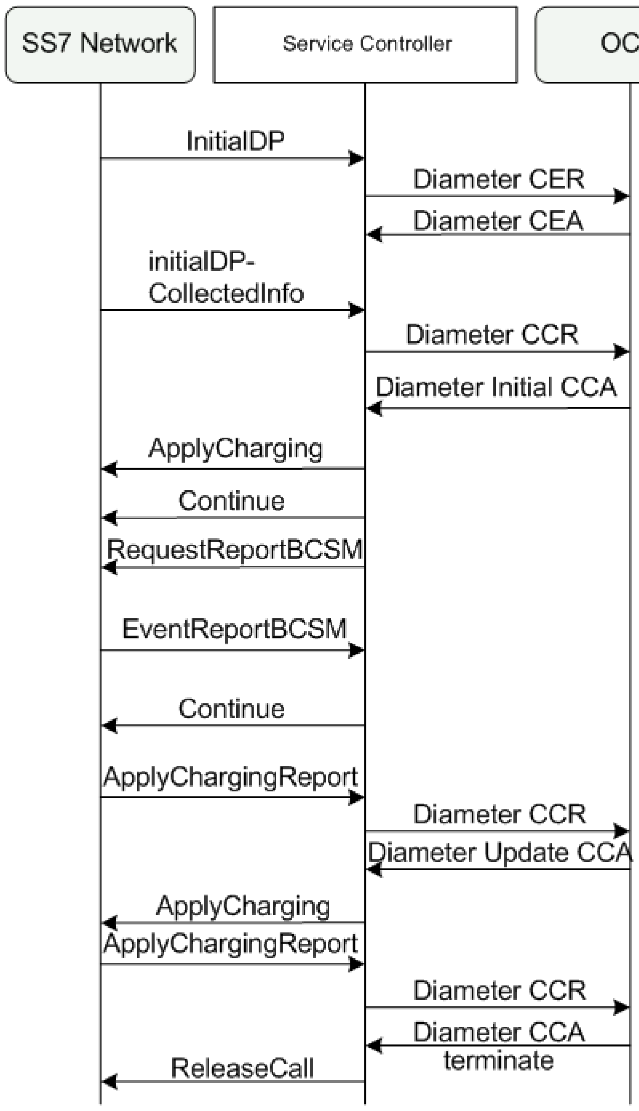


Figure 4–2 shows the components in the Service Controller processing tier that you set up to enable SS7 to OCS charging.

The IMSCF Service Controller module translates SS7 messages into an internal format used by the Orchestration Engine. You use different variants of the IMSCF module, depending on the protocol variant used in your network. For example, if your network uses CAP Phase 3, then you use the IMSCF-CAP-Phase 3 module.

To enable Service Controller's SS7 and Ro connectivity to OCS, you also need to configure the appropriate signaling server units (SSU) for your OCS in the signaling tier. The OCS SSUs are not included in the figure. See *Service Controller Signaling Server Units Configuration Guide* for more information about configuring the SSU(s) for your OCS.

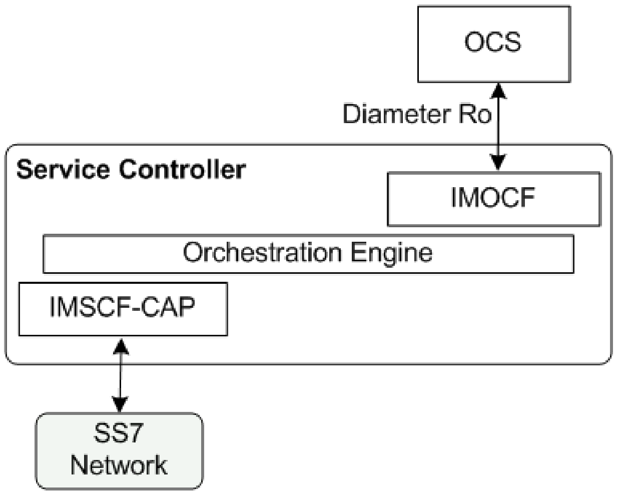
Sessions flow from the SS7 network through the components in the following order:

1. SS7 Network
2. SSU SS7
3. IMSCF

4. Orchestration Engine
5. Ro
6. OCS

Responses from the OCS flow in the reverse direction through the same components

Figure 4–2 Components Required for SS7 Session Charging



Enabling OCS Charging of SS7 Sessions

An end-to-end configuration enabling SS7 charging by a Diameter Ro OCS, requires configuration of SSU SS7, SSU Diameter, IMSCF, IMOCF, and orchestration engine rules.

To set up this configuration:

1. Enable Service Controller to accept traffic of sessions arriving from the legacy SS7 network. See "Setting Up Connectivity to the Legacy SS7 Network" for information about the components that you need to configure and how to configure them.
2. Enable the diameter Ro interface to allow communication between Service Controller and the OCS. See "Setting Up Connectivity to the OCS" for more information.
3. Route sessions that arrive from the legacy SS7 network to the OCS. See "Defining a Service Orchestration Chain" for more information.

Setting Up Connectivity to the Legacy SS7 Network

You can connect Service Controller to a SIGTRAN-based SS7 network. Service Controller does not support TDM-based SS7 networks.

Before you start, make sure that you have a detailed plan of how Service Controller connects to your SS7 network, including point codes that you assign to Service Controller. To connect Service Controller to the legacy SS7 network:

1. Configure a SIGTRAN-based SS7 SSU for your SIGTRAN-based SS7 network. See *Service Controller Signaling Server Units Configuration Guide* for more information.
2. Deploy and configure the proper IMSCF module variant, as described in *Service Controller Modules Configuration Guide*.

3. Configure routing rules in the SS7 SSU to route sessions arriving from the SS7 network to the IMSCF module that you deployed in the step 1.

   In the Administration Console:

   a. In the navigation tree, expand **OCSB** node, and then the **Signaling Tier** node.
   b. Select **SSU SS7 SIGTRAN**.
   c. In the **Routing** tab, in the left pane, click the **Add** button. The **New** dialog box appears.
   d. In the **Name** field, enter a name for the routing rule. Click **Apply**. The newly created rule now appears in the rules tree in the left pane.
   e. Select the newly created rule node, and then select the **Incoming Routing Rules** tab.
   f. In the **Module Instance** field enter *module*.*type*@*domain* where *module* is the name of the IMSCF module that you deployed in step 2, *type* is the type of the IMSCF module that you deployed, and *domain* is the name of the domain where you deployed the module. For example, for an instance of IMSCFCAP3, **imscf.IMSCFCAP3@ocsb**.
   g. Optionally, you can define criteria for the rule, so that the SSU SS7 route to the IMSCF application only sessions that meet the criteria. You define criteria for the rule in the **Incoming Routing Rule** Criteria tab.

4. Configure the IMSCF parameters under the **Charging Service** tab in the Administration Console. See the charging service configuration section for the IMSCF module you are using in *Service Controller Modules Configuration Guide*, for more information on setting these parameters.

Setting Up Connectivity to the OCS

Before you start, make sure that you have a detailed plan of how Service Controller connects to your OCS, including information such as connection credentials and IP addresses or host names. Ask your OCS administrator for the required information

You can connect Service Controller to the OCS through Diameter Ro.

To connect Service Controller to the IMS network:

1. Define Service Controller as a Diameter node and configure how other Diameter entities access it, as described in the discussion about creating Diameter nodes in the chapter about the Diameter SSU in *Service Controller Signaling Server Units Configuration Guide*.

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node, and then the **Signaling Tier** node.
   b. Select the **SSU Diameter** node.
   c. In the **DIAMETER** tab, select the **Diameter Configuration** tab.
   d. You can either use the default node or create a new node by clicking the **Add** button on the bottom of the list of existing Diameter nodes.
   e. In the **General** tab, in the **Name** field, enter a unique name for the Diameter node.
   f. In the **Realm** field, enter the realm name that other Diameter nodes use to access Service Controller.

g. In the **Port** field, enter the port number that signaling servers use to listen to Diameter traffic.

h. Leave the **Address**, **Host** and **Target** fields blank to apply the configuration to all signaling servers in the Signaling Domain and have them all provide a Diameter network channel on the same port.

i. Click **Apply**.

Note: If you run multiple signaling servers on the same physical machine, you have to define each signaling server as a different Diameter node which listens on a different port. Otherwise, the Diameter SSU running on all signaling servers using the same port will result in network traffic collisions.

2. Deploy the IMOCF-Ro module as described in the discussion on setting up IMOCF-Ro in *Service Controller Modules Configuration Guide*.

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node.

   b. Expand the **Processing Tier** node, and then the **Interworking Modules** node.

   c. Click **IM Management**.

   d. In the **IM Management** tab, click the **New** button. The New dialog box appears.

   e. From the **Type** list, select **IMOCF**.

   f. In the **Name** field, enter a module instance name. For example, rimocfro_instance.

   g. Click the **OK** button.

3. Configure the IMOCF-Ro module as described in the discussion on setting up IMOCF-Ro in *Service Controller Modules Configuration Guide*.

4. Activate the IMOCF-Ro module that you deployed and configured in steps 2 and 3.

   In the Administration Console:

   a. In the navigation tree, expand the **OCSB** node.

   b. Expand the **Processing Tier** node and then the **Interworking Modules** node.

   c. Click on **IM Management**.

   d. In the **IM Management** tab, select the IMOCF-Ro module in the table.

   e. Click the **Activate** button.

Defining a Service Orchestration Chain

To route sessions to the OCS, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain with the OCS in it.

See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

5 Setting Up Online Charging Solution for IMS Networks

This chapter describes the Oracle Communications Service Controller and Oracle Communications Online Mediation Controller online charging for IMS networks solution and how to configure Service Controller and Online Mediation Controller for this solution.

About the Online Charging Solution for IMS Networks

You use the online charging for IMS networks solution to deliver Diameter-based charging for subscriber sessions in an IMS network. Service Controller deliver SIP sessions to online charging systems (OCSs) including third-party charging systems supporting the Diameter Ro protocol.

Service Controller supports online charging for IMS networks with a connection to an OCS through the Diameter Ro protocol.

Figure 5–1 shows the basic session charging flow of the online charging for IMS networks solution.

Figure 5–1 Basic Call Flow in SIP to OCS Solution

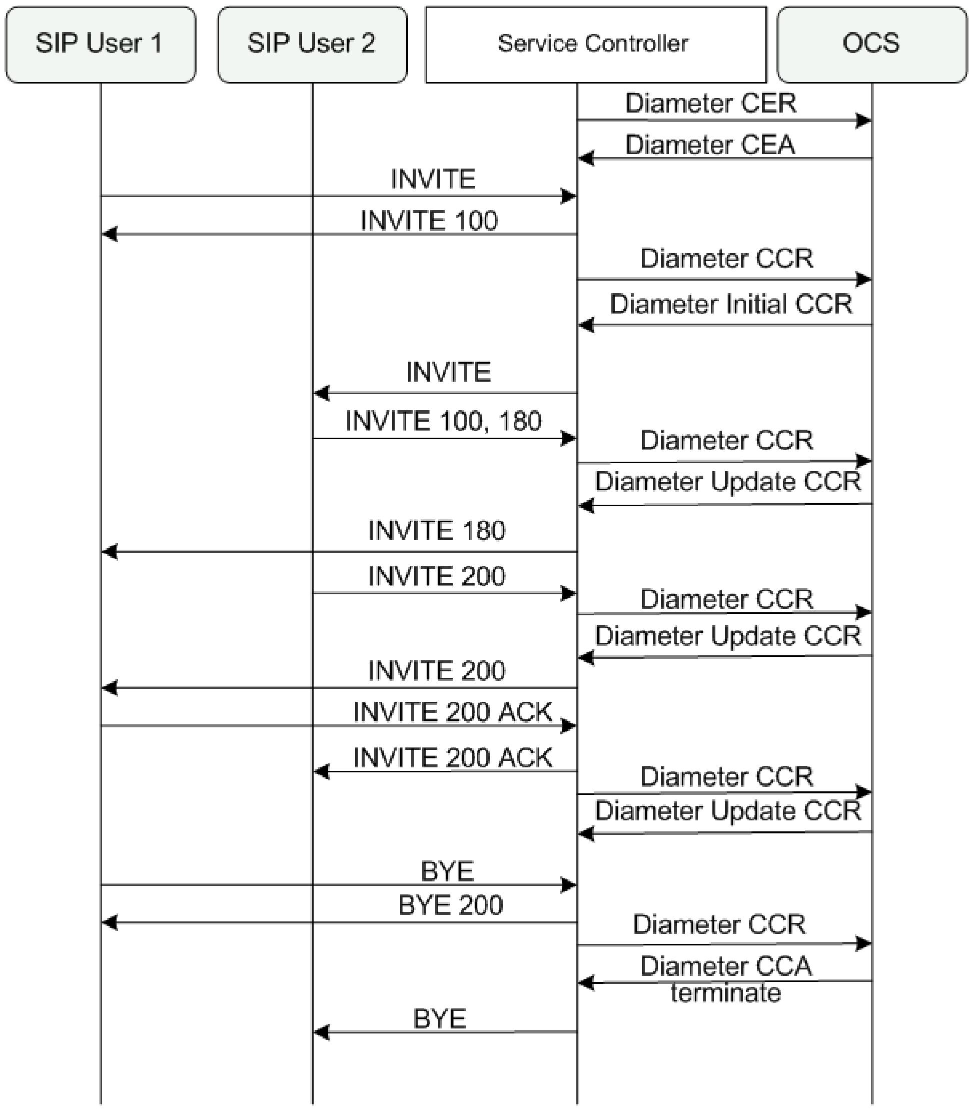


Figure 5–2 shows the components in the Service Controller and Online Mediation Controller processing tier that you set up to enable charging of SIP sessions by an OCS.

The **IMASF-SIP** Service Controller module translates the SIP messages into an internal format used by the Orchestration Engine.

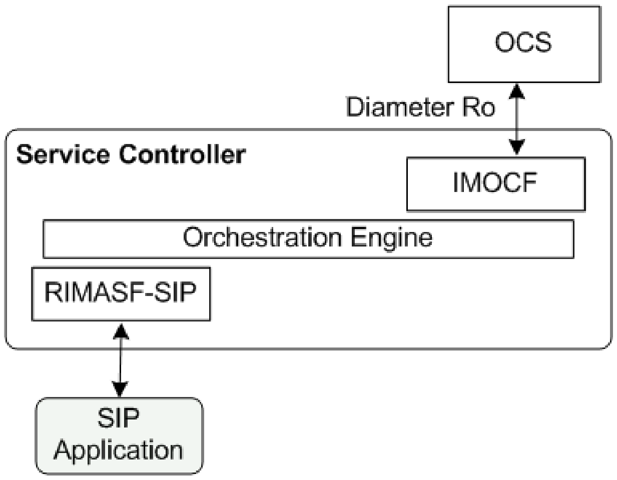
The IMOCF module translates internally formatted messages to Diameter Ro messages processed by the OCS. The SSUs are not included in the figure. See *Service Controller Signaling Server Units Configuration Guide*, for more information about configuring SSUs.

Sessions flow from the SIP applications through the components in the following order:

1. IMS Network
2. SSU SIP
3. IMASF-SIP
4. Orchestration Engine

5. Ro
6. OCS

Figure 5–2 Components Required for SIP Session Charging by OCS



Responses from the OCS flow in the reverse direction through the same components

Enabling OCS Charging of SIP Sessions

An end-to-end configuration enabling SIP charging by a Diameter Ro OCS, requires configuration of SSU SIP, SSU Diameter, IMASF, IMOCF and orchestration engine rules.

To set up this configuration:

1. Enable Service Controller to accept SIP sessions. See "Setting Up Connectivity to the IMS Network" for information about the components that you need to configure and how to configure them.
2. Install and create an Online Mediation Controller domain. See *Service Controller Installation Guide*, for more information.
3. Route sessions that arrive from the IMS network to the OCS. See "Defining a Service Orchestration Chain" for more information.

Setting Up Connectivity to the IMS Network

To enable the Service Controller SIP interface, perform the configuration steps indicated in the following sections:

- See "Enabling Acceptance of Incoming Initial SIP Messages", for information on enabling Service Controller to accept initial SIP messages.
- See "Configuring the RIMASF-SIP Module", for information on deploying the RIMASF-SIP module.
- See "Routing Incoming Initial SIP Messages to the RIMASF-SIP Module", for information on routing the initial SIP message to RIMASF-SIP

Enabling Acceptance of Incoming Initial SIP Messages

To enable Service Controller to accept incoming initial SIP messages, configure each server in the signaling tier as a SIP network access point.

In the Administration Console:

1. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
2. Select the **SSU SIP** node.
3. In the **SIP** tab, in the **SIP Configuration** tab, select the **Network Access Points** tab.
4. Click the **Add** button. The **New** dialog box appears.
5. Enter an alpha-numeric name for the newly added network access point. Click **Apply**. A new network access point appears in the tree.
6. Select the recently added network access point.
7. In the **General** tab, in the **Target** field, enter the name of the server and click **Apply**.
8. Select the **Listen Address** tab, configure the **Host** and **Port** of the network access point, and click **Apply**.
9. Select the **External Listen Address** tab. If you are using a **Load Balancer** in your system, configure the **Host** and **Port** of the Load Balancer. Otherwise, configure again the **Host** and **Port** of the network access point.

Repeat steps **4** through **9** for every server in the signaling tier.

For more details about configuring network access points in the SSU SIP, see the chapter about the SSU SIP in *Service Controller Signaling Server Units Configuration Guide*.

Configuring the RIMASF-SIP Module

Configure and deploy the RIMASF-SIP instance as described in the chapter on setting up RIMASF-SIP in *Service Controller Modules Configuration Guide*.

Routing Incoming Initial SIP Messages to the RIMASF-SIP Module

To route incoming SIP messages from your SIP applications to the RIMASF-SIP module, you configure incoming routing rules in the SSU SIP.

In the Administration Console:

1. In the navigation tree, expand the **OCSB** node, then the **Signaling Tier** node.
2. Select the **SSU SIP** node.
3. In the **SSU SIP** tab, select the **Incoming Routing Rules** tab.
4. Click the **New** button. The **New** dialog box appears.
5. In the **Name** field, enter a name for the rule.
6. In the **IP Address** field, enter the address of your SIP application.
7. In the **Alias** field, enter the alias of the RIMASF-SIP module that you want to process the messages from your SIP application. Enter ssu:*RIMASF-name*.RIMASF@*domain* where R*IMASF-name* is the name that you gave to the RIMASF module when you deployed it, and *domain* is the name of the processing domain where the module is deployed.

For more details about configuring incoming routing rules in the SSU SIP, see the chapter about the SIP SSU in *Service Controller Signaling Server Units Configuration Guide*.

Setting Up Connectivity to the OCS

See "Setting Up Connectivity to the OCS" for information on connecting Service Controller to a Diameter Ro OCS.

Defining a Service Orchestration Chain

To route sessions to the OCS, you have to choose the method (LSS, HSS, or static service orchestration) that you want to use for service orchestration, and then, based on your choice, define a service orchestration chain with the OCS in it.

See "Configuring Service Orchestration" for information about the different options for service orchestration, and how to configure them.

6

Manipulating Headers and Body of Messages

This chapter describes how to manipulate headers and the body of a message before Oracle Communications Service Controller sends it to an application.

About Manipulating Headers and the Body of the Message

Applications might impose various requirements on the structure and contents of headers and the body of messages that these applications receive from Service Controller.

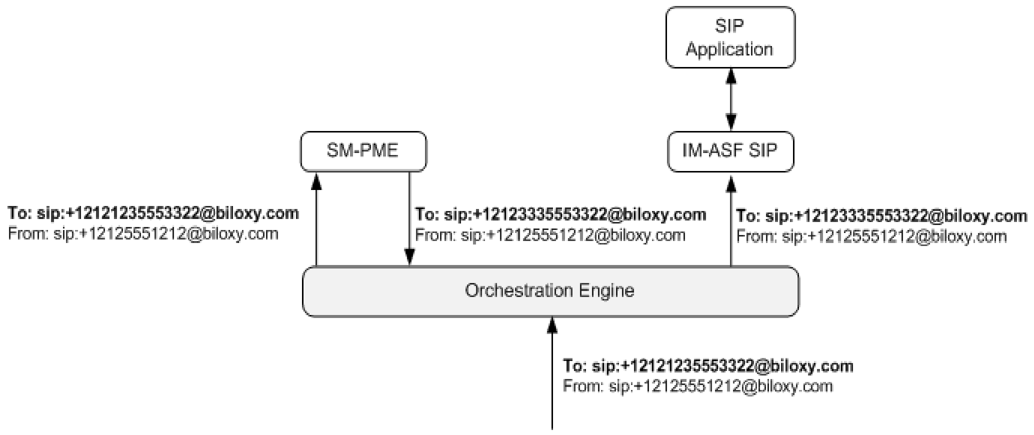
For example, the phone number prefix of a mobile operator changed. However, you want to allow mobile subscribers to use the old prefix and replace it with the new one before forwarding the call to the SIP application. In this case, you need to modify the contents of the **To** header.

Similarly, a SIP application might support only CAP phase 3 while Service Controller received the message in CAP phase 4. In this case, you might want to remove all CAP phase 4-specific headers and leaving only those headers that the SIP application can recognize.

To manipulate headers and the body of a message before sending it to the application, you add an SM-PME supplementary module to your orchestration logic.

Figure 6–1 shows an example of a session when SM-PME receives the session, modifies the value of the **To** header, and forwards the session to the SIP application.

Figure 6–1 SM-PME in the Orchestration Logic


SIP Application
SM-PME
IM-ASF SIP
To: sip:+12121235553322@biloxy.com
From: sip:+12125551212@biloxy.com
To: sip:+12123335553322@biloxy.com
From: sip:+12125551212@biloxy.com
To: sip:+12123335553322@biloxy.com
From: sip:+12125551212@biloxy.com
Orchestration Engine
To: sip:+12121235553322@biloxy.com
From: sip:+12125551212@biloxy.com

The SM-PME modifies the headers and body according to the rules that you define in a file, known as mapping file.

About the Mapping File

The mapping file is an XML file. It contains an XSL transform that the SM-PME applies to the SIP headers and body. You create the mapping file outside of Service Controller (for example, using an XML editor) and then specify the location of the mapping file when you configure the SM-PME.

If you added the SM-PME to your Service Controller deployment, but do not specify the location of the mapping file, the SM-PME forwards the session to the application without any modifications.

Converting Headers and SDP Body to the XML Representation

XSL transforms can be applied to XML-based structures only. However, headers and the body of a message are not stored in an XML. To apply the XSL transform defined in the mapping file to the headers and body, the SM-PME performs the following steps:

1. First, the SM-PME converts the headers and the body to an XML representation. This is an automatic process that does not require your involvement.
2. Then the SM-PME modifies headers and the body by applying the mapping file to the XML representation.
3. After the headers and body are modified, the SM-PME converts the headers and body from the XML representation back to their original format.

About the Structure of the Mapping File

The mapping file consists of the following sections:

- Header manipulation: This section is wrapped in the **<header>** element. A mapping file can contain only one **<header>** element.
- Body manipulation: This section is wrapped in the **<body>** element. A mapping file can contain multiple **<body>** elements. Each **<body>** element contains the **<Content-Type>** element that specifies the protocol of the body, such as SDP or CAP.

Both the header manipulation section and each body manipulation section contain an XSL transform. Example 6–1 shows how a typical mapping file is structured.

Example 6–1 Mapping File Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
   <header>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
            <!-- XSL transform for headers -->
         </xsl:stylesheet>
      ]]></xsl>
   </header>

   <body>
      <Content-Type>application/sdp</Content-Type>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
            <!-- XSL transform for the SDP body -->
         </xsl:stylesheet>
      ]]></xsl>
   </body>

   <body>
      <Content-Type>application/cap-phase4+xml</Content-Type>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
               <!-- XSL transform for the CAP4 body -->
         </xsl:stylesheet>
      ]]></xsl>
   </body>
</mapping>
```

Manipulating SIP Headers

You might need to manipulate headers of a SIP message to make the message to conform to the requirements that the SIP application imposes.

Note: The SM-PME can modify headers of SIP requests only.

Converting the Headers to XML

When the SM-PME converts a message to the XML, it represents each header as follows:

```
   <SALHeader>
      <Header>Header_Name</Header>
      <Content>Header_Content</Content>
   </SALHeader>
```

Example 6–2 shows the original format of the message headers.

Example 6–2 Original Format of SIP Message Headers

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP
bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
Via: SIP/2.0/UDP tzach.com
Max-Forwards: 69
To: sip:+12121235553322@biloxy.com
From: sip:+12125551212@biloxy.com;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
```

Example 6–3 shows the XML representation to which the SM-PME converts the headers.

Example 6–3 XML Representation of SIP Message Headers

```
<?xml version="1.0" encoding="UTF-8"?>
<SALMsgHeader>
   <Request-Line>INVITE sip:bob@biloxi.com SIP/2.0</Request-Line>
   <SALHeader>
      <Header>From</Header>
    <Content>sip:+12125551212@biloxy.com;tag=1928301774</Content>
   </SALHeader>

   <SALHeader>
      <Header>To</Header>
      <Content>sip:+12121235553322@biloxy.com</Content>
   </SALHeader>

   <SALHeader>
      <Header>Via</Header>
      <Content>"SIP/2.0/UDP
bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1"</Content>
   </SALHeader>

   <SALHeader>
      <Header>Via</Header>
      <Content>"SIP/2.0/UDP Tzach.com"</Content>
   </SALHeader>

   <SALHeader>
      <Header>Max-Forwards</Header>
      <Content>"69"</Content>
   </SALHeader>

   <SALHeader>
      <Header>Call-ID</Header>
      <Content>a84b4c76e66710</Content>
   </SALHeader>

   <SALHeader>
      <Header>CSeq</Header>
      <Content>314159 INVITE</Content>
   </SALHeader>

   <SALHeader>
      <Header>Contact</Header>
      <Content>"&lt;sip:alice @ pc33.atlanta.com&lt;"</Content>
   </SALHeader>

</SALMsgHeader>
```

Changing the Contents of the Header

Example 6–4 shows how you can code an XSL transform that changes the phone number in the **To** header from **sip:+12121235553322@biloxy.com** to **sip:+12123335553322@biloxy.com**.

The transform copies the rest of the headers without any modification. (See http://www.w3.org/TR/xslt for more information about XSLT.)

The code related to the headers manipulation is emphasized with bold.

Example 6–4 SIP Headers Manipulation

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
   <header>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
            <xsl:template match="/SALMsgHeader">
```

```
                    <SALMsgHeader>
                       <xsl:copy-of select="node()[(name()='Request-Line')]"/>
                       <xsl:copy-of select="node()[(name()='Status-Line')]"/>
                       <xsl:for-each select="SALHeader">
                             <SALHeader>
                                <xsl:choose>
                                   <xsl:when test="Header='To'">
                                      <xsl:copy-of select="Header"/>
                                      <Content>sip:+12123335553322@biloxy.com</Content>
                                  </xsl:when>
                                  <xsl:otherwise>
                                     <xsl:copy-of select="Header"/>
                                     <xsl:copy-of select="Content"/>
                                  </xsl:otherwise>
                                </xsl:choose>
                             </SALHeader>
                          </xsl:for-each>
                       </SALMsgHeader>
                    </xsl:template>
                 </xsl:stylesheet>
             ]]></xsl>
      </header>

      <body>
         <!-- XSL transform for the SDP body -->
      </body>
   </mapping>
```

Manipulating an SDP Body

The SM-PME can modify the body of both SIP requests and responses.

After you changed the SDP body, you might need to change the **Content-Type** header of the SDP body. This can be required when you converted a message from one protocol to another.

For example, the original message is in CAP phase 4. Therefore, the Content-Type header is set to **application/cap-phase4+xml**. However, the SIP application supports CAP phase 3 only. Therefore, you need to remove some headers and left only those headers which are supported in CAP phase 3. Then in the resulted message, you should change the **Content-Type** to **application/cap-phase3+xml**.

Converting the SDP Body

When the SM-PME converts an SDP body of the message to the XML, it represents each field in the SDP body as follows:

```
<Field>
   <Type>Field_Name</Type>
   <Value>Field_Value</Value>
</Field>
```

The SM-PME converts the body to the XML representation only if there is at least one non-empty **<body>** element.

Example 6–5 shows the original format of the SDP body.

Example 6–5 Original SDP Body

```
v=0
o=user1 53655765 2353687777 IN IP4 10.162.34.115
```

```
c=IN IP4 10.162.34.115
t=0 0
m=audio 6001 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Example 6–6 shows the XML representation to which the SM-PME converts the SDP body.

Example 6–6 XML Representation of the SDP Body

```
<?xml version="1.0" ?>
<SDPSessionDescription>
   <Field>
      <Type>v</Type>
      <Value>0</Value>
   </Field>

   <Field>
      <Type>o</Type>
      <Value>user1 53655765 2353687777 IN IP4 10.162.34.115</Value>
   </Field>

   <Field>
      <Type>c</Type>
      <Value>IN IP4 10.162.34.115</Value>
   </Field>

   <Field>
      <Type>t</Type>
      <Value>0 0</Value>
   </Field>

   <Field>
      <Type>m</Type>
      <Value>audio 6001 RTP/AVP 0</Value>
   </Field>

   <Field>
      <Type>a</Type>
      <Value>rtpmap:0 PCMU/8000</Value>
   </Field>
</SDPSessionDescription>
```

Manipulating the SDP Body

Example 6–7 shows how you can create an XSL transform that sets the value of the **<Type>** element to **1 0**. (See `http://www.w3.org/TR/xslt` for more information about XSLT.) The code related to the SDP body manipulation is emphasized with bold. The **<Content-Type>** element specifies that the transform is designed for SDP messages.

Example 6–7 Manipulating the SDP Body

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
   <header>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
            <!-- XSL transform for headers -->
         </xsl:stylesheet>
      ]]></xsl>
   </header>

   <body>
      <Content-Type>application/sdp</Content-Type>
      <xsl><![CDATA[
         <xsl:stylesheet version="1.0"   xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
            <xsl:template match="/SDPSessionDescription">
               <SDPSessionDescription>
                  <xsl:for-each select="Field">
                     <Field>
                        <xsl:choose>
                           <xsl:when test="Type='t'">
                              <xsl:copy-of select="Type"/>
                              <Value>1 0</Value>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:copy-of select="Type"/>
                              <xsl:copy-of select="Value"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </Field>
                  </xsl:for-each>
               </SDPSessionDescription>
            </xsl:template>
         </xsl:stylesheet>
      ]]></xsl>
   </body>
</mapping>
```

Changing the Content Type in the Transformation Result

You can change the content type of a body in the transformation result using the **<Content-Type-Result>** element. If you do not specify this element, then the content type of the body in the transformation result is the same as in the original message.

Example 6–8 shows how you can create an XSL transform that changes the content body from **application/cap-phase4+xml** (see the value of the **<Content-Type>** element) to **application/cap-phase3+xml** (see the value of the **<Content-Type-Result>** element).

Example 6–8 Changing the Content Type

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
   <header>
      <!-- XSL transform for the SDP body -->.
   </header>

<body>
   <Content-Type>application/cap-phase4+xml</Content-Type>
   <Content-Type-Result>application/cap-phase3+xml</Content-Type-Result>
   <xsl><![CDATA[
      <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
         <xsl:template match="text()" />
            <xsl:template match="Cap4">
```

```
                <xsl:apply-templates />
            </xsl:template>
            <xsl:template match="initialDP">
                <Cap3>
                    <initialDP>
                        <xsl:copy-of select="serviceKey"/>
                        <xsl:copy-of select="bearerCapability"/>
                        <xsl:copy-of select="timeAndTimezone"/>
                    </initialDP>
                </Cap3>
            </xsl:template>
        </xsl:stylesheet>
    ]]></xsl>
</body>
</mapping>
```

Setting Up Manipulation of Message Headers and Body

To set up manipulation of message headers and body:

1. Create a mapping file.
2. Add an instance of SM-PME to your Service Deployment. See the discussion on adding an SM-PME to the Service Controller deployment in *Service Controller Modules Configuration Guide* for more information.
3. Specify the location of the mapping file and define other parameters in the configuration of the SM-PME. See the discussion on setting up an SM-PME in *Service Controller Modules Configuration Guide* for more information.
4. Add the SM-PME to the orchestration logic. See the discussion on building an orchestration logic in *Service Controller Orchestration User's Guide*.

7 Configuring Service Orchestration

This chapter describes how to configure service orchestration in Oracle Communications Service Controller.

About Service Orchestration

Service Controller can invoke one or more applications, combining and delivering multiple services to sessions in the network. Service orchestration is performed by the Orchestration Engine based on an orchestration logic which defines how to route sessions through a number of applications. The Orchestration Engine invokes the applications in a particular order, according to conditions in the orchestration logic that determine which applications to invoke and in what order.

To create the orchestration logic you use the Orchestration Studio. See *Service Controller Orchestration User's Guide* for information about the Orchestration Studio and how to use it to define an orchestration logic.

After you create the orchestration logic, you store it as part of the subscriber's profile. Whenever Service Controller handles a session, it first obtains the profile of the subscriber that owns the session, and then route the session according to the orchestration logic in that profile. Service Controller support the following subscriber profile stores:

- Local Subscriber Server (LSS), which is an on-board implementation of a profile server. The LSS is capable of storing subscriber profiles, including orchestration logic given in the Initial Filter Criteria (iFC) format. The LSS is implemented as a supplementary module named SM-LSS. See the chapter about SM-LSS in *Service Controller Modules Configuration Guide* for information about configuring the LSS.
- Home Subscriber Server (HSS), which is the standard primary user database in the IMS domain. Service Controller uses a standard Diameter Sh interface to connect the HSS obtain the orchestration logic.

When you configure Service Controller you choose only one of the subscriber profile store options. You define which subscriber store you use in the Orchestration Engine, and configure fields in the Orchestration Engine configuration screens that are relevant to the subscriber store of your choice.

It is possible to configure the Orchestration Engine to route all sessions through one or more applications that you define, without using an orchestration logic, and without going to the LSS or HSS to obtain the orchestration logic. See "Using Static Service Orchestration" for more information.

Using Static Service Orchestration

When you configure a static service orchestration, Service Controller routes all session through one or more applications that you define, without using any orchestration logic, and without turning to a subscriber profile store, such as LSS and HSS, to obtain the orchestration logic.

To configure a static service orchestration, in the Administration Console:

1. In the navigation tree, expand the **OCSB** node.
2. Expand the **Processing Tier** node, and then the **Orchestration Engine** node.
3. In the **Configuration** tab, select the **General** tab.
4. In the **Subscriber Profile Receiver** list, select **OlpDefaultInfoReceiver**.
5. In the **Configuration** tab, select the **Static Route OLP** tab.
6. In the **Default Routing Targets** field, enter a list of alpha-numeric addresses of target applications. For each application enter sip:*module.type*@*example*.com, where *module* is the name of the interworking module that enables the connection with the target application, *type* is type of the module, and *example* is the domain name.

   For example, sip:imasfsip.IMASF@*example*.com

   Use the space character to separate between two target applications in the list.

Using Orchestration Logic in the LSS for Service Orchestration

When you choose LSS to be your profile database, you create the service orchestration logic using the Orchestration Studio, store the orchestration logic under the subscriber profile in the LSS, and configure the Orchestration Engine to obtain orchestration logic from the LSS.

To configure orchestration logic in the LSS for Service Orchestration:

1. Define a new profile in the LSS:
   a. In the Administration Console, in the navigation tree, expand the **OCSB** node.
   b. Expand the **Processing Tier** node, and then the **Supplementary Modules** node.
   c. Select the **SM-LSS** node.
   d. In the **Configuration** tab, select the **Orchestration Profiles** tab.
   e. Click **New** button. The New dialog box appears.
   f. Enter the fields in the New dialog box as described in the section about configuring orchestration profiles, in the chapter about configuring SM-LSS, in *Service Controller Modules Configuration Guide*.
   g. Click **Apply**.
2. Use the Orchestration Studio to define the orchestration logic for the new profile that you defined in step 1. In the Orchestration Studio, from the **iFC** list, select the name of the profile. Define an orchestration logic and save it. See *Service Controller Orchestration User's Guide* for information about using the Orchestration Studio to define an orchestration logic.
3. Configure the Orchestration Engine to use the LSS for its subscriber profile store.
   a. In the Administration Console, in the navigation tree, expand the **OCSB** node.

b. Expand the **Processing Tier** node, and then select the **Orchestration Engine** node.

c. In the **Configuration** tab, select the **General** tab.

d. In the **Subscriber Profile Receiver** list, select **OlpLSSInfoReceiver**.

Using Orchestration Logic in an HSS for Service Orchestration

When you choose an external HSS for your profile database, you configure the Orchestration Engine to use the service orchestration logic stored under the profiles in the HSS.

To configure the Orchestration Engine to use the HSS for its subscriber profile store:

1. In the Administration Console, in the navigation tree, expand the **OCSB** node.
2. Expand the **Processing Tier** node, and then select the **Orchestration Engine** node.
3. In the **Configuration** tab, select the **General** tab.
4. In the **Subscriber Profile Receiver** list, select **OlpHSSInfoReceiver**.
5. In the **Configuration** tab, select the **HSS OLP** tab.
6. The interface with the HSS is a standard Diameter Sh interface. Use the **Destination-Host AVP** and **Destination-Realm AVP** fields to configure the address of the HSS.

   If you previously defined the HSS as an outbound destination in the SSU Diameter, you can enter the alias of the outbound destination in the **Destination-Host AVP** field.

To provision orchestration logic in the HSS, you use tools provided with the HSS. To define orchestration logic that you later provision in the HSS, you can use the Orchestration Studio. In the Orchestration Studio, from the **iFC** list, select **default**. Use the Orchestration Studio's tools to define and edit the orchestration logic. Once the orchestration logic is ready, you can use the generated iFC to provision it in the HSS.

Updating the Service Orchestration Chain

After you create an orchestration chain and store it, you can update it to add applications to the chain, remove applications from the chain, change application invocation criteria and so on.

The procedure for updating the service orchestration chain depends on the method that you use for service orchestration; that is, static service orchestration, orchestration logic in the LSS, or orchestration logic in the HSS.

You can use the Service Controller graphical user interface to change a static service orchestration or the orchestration logic stored in the LSS. If you use HSS to store orchestration logics, you need to use the tools provided with the HSS to change the orchestration logic that it stores.

Updating the Static Orchestration Chain

To add or remove an application from the a static orchestration chain:

1. In the Administration Console, in the navigation tree, expand the **OCSB** node.
2. Expand the **Processing Tier** node, and then select the **Orchestration Engine** node.
3. In the **Configuration** tab, select the **Static Route OLP** tab.

4. In the **Default Routing Targets** field, either add the alpha-numeric address of an application to the list of applications, or remove an address from the list. If you add an address, enter sip:*module.type*@*example*.com, where *module* is the name of the module that enables the connection with the target application, *type* is type of the module, and *example* is the domain name.

Updating the Orchestration Logic in the LSS

To update the orchestration logic of a profile stored in the LSS, in the Orchestration Studio, from the **iFC** list, select the name of the LSS profile whose orchestration logic you want to change. Update the orchestration logic and save it.

Updating the Orchestration Logic in the HSS

To update the orchestration chain stored in the HSS, use the tools provided with the HSS to change the orchestration logics. Alternatively, you can use the default orchestration profile in the Orchestration Studio to define the updated orchestration logic. Then, to provision the updated orchestration logic in the HSS, you use tools provided by the HSS.

8 Monitoring Service Controller

This chapter describes how to monitor Oracle Communications Service Controller.

About Monitoring Service Controller

You can monitor how Service Controller operates by receiving the following information:

- Statistics on messages and sessions that Interworking Modules (IMs) and the Orchestration Engine (OE) handle. See "Monitoring the Processing Domain" for more information.
- Status of the network entities with which Signaling Server Units (SSUs) communicate. See "Monitoring the Signaling Domain" for more information.
- Availability of the SIP ports.

Monitoring the Processing Domain

A deployment of Service Controller might involve the following components:

- Orchestration Engine
- IN interface:
  - IM-SCF (CAP1-4, INAP CS-1, WIN, AIN)
  - IM-SSF (CAP1-3, INAP CS-1, WIN, AIN)
  - IM-PSX MAP
- SS7 interface:
  - IM-SCF (CAP1-4, INAP CS-1, WIN, AIN)
  - R-IM-SCF (CAP1-3, INAP CS-1, WIN, AIN)
- SIP interface:
  - IM-ASF SIP
  - R-IM-ASF SIP
- SMPP interface:
  - IM-UIX-USSD SMPP
  - IM-UIX-SMS SMPP

Using runtime MBeans, you can gather statistics on sessions and messages that these IMs send and receive through their interfaces.

Monitoring the IN Interface

IM-SCF, IM-SSF, and IM-PSX MAP modules provide the IN interface. You can get counters of the following:

- Dialogs initiated by the network and IM. See "Getting Counters of Dialogs" for more information.
- Messages that the IM handled. See "Getting Counters of Messages" for more information.

Getting Counters of Dialogs

Using **DialogRuntimeMBean**, you can get the counters of dialogs initiated by the IM and the network.

Service Controller creates a separate instance of DialogRuntimeMBean for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is **com.convergin:Type=DialogRuntime,Version=**
*MBean_Version***,Location=**<*server-name*>**,Name=***IM_Instance_Name***.Dialog**.*protocol*

Table 8–1 describes the counters that **DialogRuntimeMBean** provides.

Table 8–1 DialogRuntimeMBean Counters

To Get Total Number of...	Use...
Dialogs initiated by the network	NetworkInitiatedDialogCount
Dialogs initiated by the IM	ServiceInitiatedDialogCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Counters of Messages

Using **MessageRuntimeMBean**, you can get the counters of messages that the IM sent and received.

Service Controller creates a separate instance of **MessageRuntimeMBean** for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is
com.convergin:Type=MessageRuntime,Version=<*version*>**,Location=**<*server-name*>**,N
ame=**<*module-instance-name*>**.Message.MAP**

Table 8–2 describes the counters that **MessageRuntimeMBean** provides.

Table 8–2 MessageRuntimeMBean Counters

To Get Total Number of...	Use...
Messages that the IM sent to the network	SndCount
Messages that the IM received from the network	RcvCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on the TCAP Interface

All IM-SCF and IM-SSF modules provide the TCAP interface. You can get statistics on the following:

- TCAP transactions. See "Getting Counters of TCAP Transactions" for more information.
- TCAP Abort messages. See "Getting Counters of TCAP Abort Messages" for more information.
- TCAP messages. See "Getting Counters of TCAP Messages" for more information.
- TCAP components. See "Getting Counters of TCAP Components" for more information.

Getting Counters of TCAP Transactions

Using **TcapRuntimeMBean**, you can get the counters of TCAP transactions.

Service Controller creates a separate instance of **TcapRuntimeMBean** for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is **com.convergin:Type=TcapRuntime,Version=**<*version*>**,Location=**<*server-name*>**, Name=**<*module-instance-name*>**.Tcap**.

Table 8–3 describes the counters that **TcapRuntimeMBean** provides.

Table 8–3 TcapRuntimeMBean Counters

To Get Total Number of...	Use...
Opened TCAP Transactions initiated by the application	AppInitiatedTransCount
TCAP transactions that have been closed on the IM	DestroyTransactionCount
Opened TCAP transactions initiated by the user	NetworkInitiatedTransCount
TCAP transactions that the IM handled	TransactionCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Counters of TCAP Abort Messages

Using **TcapPAbortCountMBean**, you can get the counters of TCAP Abort messages.

Service Controller creates a separate instance of **TcapPAbortCountMBean** for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is **com.convergin:Type=TcapPAbortCountRuntime,Version=**<*version*>, **Location=**<*server-name*>**,Name=**<*module-instance-name*>**.TcapPAbortCount**

Table 8–4 describes the counters that **TcapPAbortCountMBean** provides.

Table 8–4 TcapPAbortCountMBean Counters

To Get Total Number of...	Use...
P-ABORT messages received from the network with the Reason header set to Abnormal	NoCommonDialoguePortionCount
P-ABORT messages received from the network with the Reason header set to BADLY_STRUCTURED_DIALOGUE_PO RTION	BadlyFormattedTransactionPortionCount
TC-P-ABORT messages received from the network with the Reason header set to Incorrect Transaction Portion	IncorrectTransactionPortionCount
TC-P-ABORT messages received from the network with the Reason header set to INCONSISTENT_DIALOGUE_PORTION	NoCommonDialoguePortionCount
TC-P-ABORT messages received from the network with the Reason header set to Resource Limit	ResourceLimitationCount
TC-P-ABORT messages received from the network with the Reason header set to Unrecognized Message Type	UnrecognizedMessageTypeCount
TC-P-ABORT messages received from the network with the Reason header set to 'Unrecognized Transaction ID	UnrecognizedTransactionIDCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Counters of TCAP Messages

Using **TcapMessageCountMBean**, you can get the counters of the following messages:

- UniFromApplication
- UniFromNetwork
- BeginFromApplication
- BeginFromNetwork
- ContinueFromApplication
- ContinueFromNetwork
- EndFromApplication
- EndFromNetwork
- UAbortFromApplication
- UAbortFromNetwork
- Notice

Service Controller creates a separate instance of **TcapMessageCountRuntimeMBean** for each direction of each message for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is **com.convergin:Type=TcapMessageCountRuntime,Version=**<*version*>**,Location=**<*server-name*>,**Name=**<*module-instance-name*>.<*tcap-message*>

Table 8–5 describes the counters that TcapMessageCountMBean provides.

Table 8–5 TcapMessageCountMBean Counters

To Get Total Number of...	Use...
Messages that the IM received but failed to process successfully	ReceiveFailedCount
Messages that the IM received and processed successfully	ReceiveSuccessCount
Messages that the IM failed to send to the SS7 network	SentFailedCount
Messages that the IM successfully sent to the SS7 network	SentSuccessCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Counters of TCAP Components

Using **TcapComponentRuntimeMBean**, you can get the counters of TCAP components.

Service Controller creates a separate instance of **TcapComponentRuntimeMBean** for each instance of IM-SCF, IM-SSF, and IM-PSX MAP.

The object name of this MBean is **com.convergin:Type=TcapComponentRuntime,Version=**<*version*>**,Location=**<*server-name*>,**Name=**<*module-instance-name*>.TcapComponent

Table 8–6 describes the counters that **TcapComponentRuntimeMBean** provides.

Table 8–6 TcapComponentRuntimeMBean Counters

To Get Total Number of...	Use...
Invoke component timer expiry count	ComponentInvokeTimerExpiryCount
Reject component timer expiry count	ComponentRejectTimerExpiryCount
TCAP Error components that the IM received from the network	ReceivedErrorCount
TCAP Invoke components that the IM received from the network	ReceivedInvokeCount
TCAP ResultL components that the IM received from the network	ReceivedResultLCount
TCAP ResultNL components that the IM received from the network	ReceivedResultNLCount
received TCAP TimerReset components that the IM received from the network	ReceivedTimerResetCount
TCAP UCancel components that the IM received from the network	ReceivedUCancelCount
U-REJECT components that the IM received from the network	ReceivedURejectCount

Table 8–6 (Cont.) TcapComponentRuntimeMBean Counters

To Get Total Number of...	Use...
TCAP Error components that the IM sent to the network	SentErrorCount
TCAP Invoke components that the IM sent to the network	SentInvokeCount
TCAP LReject components that the IM sent to the network	SentLRejectCount
TCAP ResultNL components that the IM sent to the network	SentResultNLCount
TCAP RReject components that the IM sent to the network	SentRRejectCount
TCAP LCancel components that the IM sent to the network	SentLCancelCount
TCAP ResultL components that the IM sent to the network	SentResultLCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-SCF

In addition to the TCAP interface, you can get the counters of sessions that the IM-SCF handles on a specific interface. Depending on the interface that you want to monitor, different runtime MBeans should be used.

Table 8–7 describes the IM-SCF counters.

Table 8–7 IM-SCF Counters

To Get Total Number of...	Use...	In the Runtime MBean…
Fully controlled sessions that the IM-SCF handled in the last measurement period. Full control means that the IM-SCF handled the session during the session setup and continues to handle this session after the session is established	FullControlCount	CAP: ImscfCapRuntimeMBean WIN: ImscfWinRuntimeMBean CS-1: ImscfInapCs1RuntimeMBean AIN: ImscfAinRuntimeMBean
Non-fully controlled sessions. Non-full control means that the IM-SCF handled the session only during the session setup.	InitialControlCount	CAP: ImscfCapRuntimeMBean WIN: ImscfWinRuntimeMBean CS-1: ImscfInapCs1RuntimeMBean AIN: ImscfAinRuntimeMBean
Sessions that the IM-SCF handled	SessionCount	CAP: ImscfCapRuntimeMBean WIN: ImscfWinRuntimeMBean CS-1: ImscfInapCs1RuntimeMBean AIN: ImscfAinRuntimeMBean

Table 8–7 (Cont.) IM-SCF Counters

To Get Total Number of...	Use...	In the Runtime MBean…
Messages that the IM-SCF received from the network	RcvCount	MessageByOpRuntimeMBean InitialDpByEventTypeRuntimeMBean (for CAP and CS-1 only) ErbByEventTypeRuntime MBean (for CAP and CS1 only) InitialDpByServiceKeyRuntimeMBean (for CAP and CS1 only) MessageRuntimeMBean
Messages that the IM-SCF sent to the network	SndCount	MessageByOpRuntimeMBean InitialDpByEventTypeRuntimeMBean (for CAP and CS-1 only) ErbByEventTypeRuntime MBean (for CAP and CS-1 only) InitialDpByServiceKeyRuntimeMBean (for CAP and CS-1 only) MessageRuntimeMBean

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-SSF

In addition to the TCAP interface, you can get the counters of sessions that the IM-SSF handles on a specific interface. Depending on the interface that you want to monitor, different runtime MBeans should be used.

Table 8–8 describes the IM-SSF counters.

Table 8–8 IM-SSF Counters

To Get Total Number of...	Use...	In the Runtime MBean…
Fully controlled sessions that the IM-SSF handled in the last measurement period. Full control means that the IM-SSF handled the session during the session setup and continues to handle this session after the session is established	FullControlCount	CAP: ImssfCapRuntimeMBean WIN: ImssfWinRuntimeMBean AIN: ImssfAinRuntimeMBean
Non-fully controlled sessions. Non-full control means that the IM-SSF handled the session during the session setup only.	InitialControlCount	CAP: ImssfCapRuntimeMBean WIN: ImssfWinRuntimeMBean AIN: ImssfAinRuntimeMBean
Sessions that the IM-SSF handled	SessionCount	CAP: ImssfCapRuntimeMBean WIN: ImssfWinRuntimeMBean CS-1: ImssfInapCs1RuntimeMBean AIN: ImssfAinRuntimeMBean

Table 8–8 (Cont.) IM-SSF Counters

To Get Total Number of...	Use...	In the Runtime MBean…
Messages that the IM-SSF received from the network	RcvCount	MessageByOpRuntimeMBean InitialDpByEventTypeRuntimeMBean (CAP and CS-1 only) ErbByEventTypeRuntimeMBean (CAP and CS-1 only) InitialDpByServiceKeyRuntimeMBean (CAP and CS-1 only) MessageRuntimeMBean
Messages that the IM-SSF sent to the network	SndCount	MessageByOpRuntimeMBean InitialDpByEventTypeRuntimeMBean (CAP and CS-1 only) ErbByEventTypeRuntime MBean (CAP and CS-1 only) InitialDpByServiceKeyRuntimeMBean (CAP and CS-1 only) MessageRuntimeMBean

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Monitoring the SIP Interfaces

You can gather statistics on the SIP interface by using counters provided by IM-ASF SIP and R-IM-ASF SIP. In addition to IM-specific counters, both these modules also have the SIP interface which is common for all IM-ASF and R-IM-ASF modules.

Getting Statistics on the SIP Interface

You can get the counters of sessions that the IM-ASF or R-IM-ASF handles on the SIP interface.

Table 8–9 describes attributes that enable you to monitor the SIP interface.

Table 8–9 SIP Interface Counters

To Get Total Number of...	Use...	In Runtime MBean...
SIP sessions that the IM handled	SessionCount	SipRuntimeMBean
SIP requests that the IM received from the network	RcvCount	SipRequestRuntime MBean SipRequestByMethod RuntimeMBean SipResponseByRequest MethodRuntimeMBean
SIP requests that a module sent to the network	SndCount	SipRequestRuntime MBean SipRequestByMethod RuntimeMBean SipResponseByRequest MethodRuntimeMBean

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-ASF SIP

Using **ImasfSipRuntimeMBean**, you can get the counter of sessions that the IM-ASF SIP handles.

Service Controller creates a separate instance of **ImasfSipRuntimeMBean** for each instance of IM-ASF SIP.

The object name of this MBean is **com.convergin:Type=ImasfSipRuntime,Version=**<*version*>**,Location=**<*server-name*>, **Name=**<*module-instance-name*>.ImasfSip

Table 8–10 describes the counter that **ImasfSipRuntimeMBean** provides.

Table 8–10 ImasfSipRuntimeMBean Counter

To Get Total Number of...	Use...
Sessions that the IM-ASF SIP handled	SessionCount

For more information on how to access runtime MBeans, see a discussion on monitoring Service Controller using Runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on R-IM-ASF

Using **RimasfSipRuntimeMBean**, you can get the counter of sessions that R-IM-ASF SIP handles.

Service Controller creates a separate instance of **RimasfSipRuntimeMBean** for each instance of R-IM-ASF SIP.

The object name of this MBean is **com.convergin:Type=ImasfSipRuntime,Version=**<*version*>**,Location=**<*server-name*>, **Name=**<*module-instance-name*>.RimasfSip

Table 8–11 describes the counter that **RimasfSipRuntimeMBean** provides.

Table 8–11 RimasfSipRuntimeMBean Counter

To Get Total Number of...	Use...
Sessions that the R-IM-ASF SIP handled	SessionCount

For more information on how to access runtime MBeans, see a discussion on monitoring Service Controller using Runtime MBeans in *Service Controller System Administrator's Guide*.

Monitoring the SMPP Interfaces

You can gather statistics on the SMPP interfaces by using counters provided by IM-UIX-SMS SMPP and IM-UIX-USSD SMPP.

Getting Statistics on the SMPP Interface

Both IM-UIX-SMS and IM-UIX-USSD use the SMPP interface. Using **SmppRuntimeMBean**, you can get counters of sessions that the IM sent and received through this interface.

Service Controller creates a separate instance of **SmppRuntimeMBean** for each instance of IM-UIX-SMS and IM-UIX-USSD.

The object name of this MBean is **com.convergin:Type=SmppRuntime,Version=**<*version*>**,Location=**<*server-name*>**, Name=**<*module-instance-name*>**.Smpp**

Table 8–12 describes the counters that **SmppRuntimeMBean** provides.

Table 8–12 SmppRuntimeMBean Counters

To Get Total Number of...	Use...
Sessions that the IM-UIX-SMS handled	SessionCount
Requests that the IM-UIX-SMS handled	RequestCount
Responses that the IM-UIX-SMS handled	ResponseCount
submit_sm operations that the IM-UIX-SMS handled	SubmitSmCount
deliver_sm operations that the IM-UIX-SMS handled	DeliverSmCount
submit_sm_resp operations that the IM-UIX-SMS handled	SubmitSmRespCount
deliver_sm_resp operations that the IM-UIX-SMS handled	DeliverSmRespCount
submit_sm success responses that the IM-UIX-SMS handled	SubmitSmRespSuccessCount
submit_sm error responses that the IM-UIX-SMS handled	SubmitSmRespErrorCount
deliver_sm success responses that the IM-UIX-SMS handled	DeliverSmRespSuccessCount
deliver_sm error responses that the IM-UIX-SMS handled	DeliverSmRespErrorCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-UIX-SMS SMPP

Using **ImUixSmsSmppRuntimeMBean**, you can get the counters of sessions that the IM-UIX-SMS SMPP handles.

Service Controller creates a separate instance of **ImUixSmsSmppRuntimeMBean** for each instance of IM-UIX-SMS SMPP.

The object name of this MBean is **com.convergin:Type=ImuixSmsRuntime,Version=**<*version*>**,Location=**<*server-name*>, **Name=**<*module-instance-name*>**.ImuixSms**

Table 8–13 describes the counters that **ImUixSmsSmppRuntimeMBean** provides.

Table 8–13 ImUixSmsSmppRuntimeMBean Counters

To Get Total Number of...	Use...
SMS sessions initiated by the network.	NetworkInitiatedSessionCount
SMS sessions initiated by the application	ApplicationInitiatedSessionCount
SMSC delivery receipts which are carried over deliver_sm operations, that IM-UIX-SMS received	SmscDeliveryReceiptRcvCount
delivery_sm operations with the Message Type parameter set to 0	SmsOverDeliverSmRcvCount
Delivery ACK messages which are carried over deliver_sm operations, that IM-UIX-SMS received from an SME	SmeDeliveryAckRcvCount
User/Manual ACK messages which are carried over deliver_sm operations, that IM-UIX-SMS received from an SME	SmeUserManualAckRcvCount

For more information on how to access runtime MBeans, see a discussion on monitoring Service Controller using Runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-UIX-USSD SMPP

Using **ImuixUssdRuntimeMBean**, you can get the counters of sessions that the IM-UIX-USSD SMPP handles.

Service Controller creates a separate instance of ImuixUssdRuntimeMBean for each instance of IM-UIX-USSD SMPP.

The object name of this MBean is **com.convergin:Type=ImuixUssdRuntime,Version=**<*version*>**,Location=** <*server-name*>,**Name=**<*module-instance-name*>**.ImuixUssd**

Table 8–14 describes the counters that ImuixUssdRuntimeMBean provides.

Table 8–14 ImuixUssdRuntimeMBean Counters

To Get Total Number of...	Use...
Sessions initiated by the mobile subscriber that the IM handled	MSInitiatedSessionCount
Sessions initiated by the application server that the IM handled	ServiceInitiatedSessionCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on IM-PSX Plugin

Using **ImpsxPluginRuntimeMBean**, you can get the counters of sessions that the IM-PSX Plugin handles.

Service Controller creates a separate instance of **ImpsxPluginRuntimeMBean** for each instance of IM-PSX Plugin.

The object name of this MBean is **com.convergin:Type=ImpsxPluginRuntime,Version=**<*version*>**,Location=**<*server-name*>**,Name=**<*module-instance-name*>**.ImpsxPlugin**.

Table 8–15 describes the counters that ImpsxPluginRuntimeMBean provides.

Table 8–15 ImpsxPluginRuntimeMBean Counters

To Get Total Number of...	Use...
Sessions that the IM handled	getSessionCount()
Sessions initiated by application server that the IM handled	getASInitiatedSessionCount()
Sessions initiated by the network that the IM handled	getNetworkInitiatedSessionCount()

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Getting Statistics on the Orchestration Engine

Using **OeRuntimeMBean**, you can get the counters of sessions and application triggers.

The object name of this MBean is **com.convergin:Type=OeRuntime,Version=***MBean_Version***,Location=***Server_Name*,**Name=***IM_instance_name***.Oe**

Table 8–16 describes the counters that **OeRuntimeMBean** provides.

Table 8–16 OeRuntimeMBean Counters

To Get Total Number of...	Use...
Sessions that the OE handled	SessionCount
Successful application triggering that the OE performed	SuccessfulApplicationTriggeringCount
Unsuccessful application triggering that the OE attempted to perform	UnsuccessfulApplicationTriggeringCount
2xx and 3xx responses that the OE received	SuccessfulUAServerTriggeringCounter

Using **OlpRuntimeMBean**, you can get the counter of triggering a specific Orchestration Login Processor (OLP).

The object name of this MBean is **com.convergin:Type=OlpRuntime,Version=***MBean_Version***,Location=***Server_Name***,Name=***IM_Instance_Name*.**Olp.***Olp_Name*

Table 8–17 describes the counter that **OlpRuntimeMBean** provides.

Table 8–17 OlpRuntimeMBean Counter

To Get Total Number of...	Use...
Times that the OE triggered the specific OLP	ExecutionCount

Using **OprRuntimeMBean**, you can get the counter of queries that the Orchestration Logic Processor (OLP) executed.

The object name of this MBean is **com.convergin:Type=OprRuntime,Version=** *MBean_Version***,Location=***Server_Name***,Name=***IM_Instance_Name***.Opr.***Opr_Name*

Table 8–18 describes the counters that **OprRuntimeMBean** provides.

Table 8–18 OprRuntimeMBean Counters

To Get Total Number of...	Use...
Successful queries that an OPR executed	SuccessfulQueryCount
Unsuccessful queries that an OPR attempted to execute	UnsuccessfulQueryCount

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Monitoring the Signaling Domain

A deployment of Service Controller might involve the following SSUs:

- SS7 SSU. See "Checking the Status of SS7 Network Entities" for more information.
- SMPP SSU. See "Checking the Status of SMPP Network Entities" for more information.
- SIP SSU. See "Checking the Status of SIP Network Entities" for more information.

Using runtime MBeans, you can check whether the network entity with which the SSU communicates is active.

Checking the Status of SS7 Network Entities

Using **SsuLocalPointCodeRuntimeMBean**, you can get the status of the local point code. Table 8–19 describes the attributes that **SsuLocalPointCodeRuntimeMBean** provides.

Table 8–19 SsuLocalPointCodeRuntimeMBean Attributes

Attribute	Description
getValue()	Specifies the address of the network entity in the URI format where with the colon character (:) is replaced with the underscore.
getStatus()	Specifies the point code status. Possible values: ■ 1 - Inaccessible ■ 2 - Congested ■ 3 - Accessible

Using **SsuLocalSubSystemRuntimeMBean**, you can get the status of the subsystem. Table 8–20 describes the attributes that **SsuLocalSubSystemRuntimeMBean** provides.

Table 8–20 SsuLocalSubSystemRuntimeMBean Attributes

Attribute	Description
getSSN()	Specifies the subsystem number
getInService()	Specifies a subsystem status. Possible values: ■ True: the Subsystem is in service ■ False: the Subsystem is out of service

Using **SsuRemoteSubSystemRuntimeMBean**, you can get the status of the remote subsystem. Table 8–21 describes the attributes that **SsuRemoteSubSystemRuntimeMBean** provides.

Table 8–21 SsuRemoteSubSystemRuntimeMBean Attributes

Attribute	Description
getValue()	Specifies the address of the network entity in the URI format where with the colon character (:) is replaced with the underscore.
getStatus()	Specifies a remote subsystem status. Possible values: ■ 0 - Remote subsystem is unavailable ■ 1 - Remote subsystem is available

Using **SsuRemotePointCodeRuntimeMBean**, you can get the status of the remote point code. Table 8–22 describes the attributes that **SsuRemotePointCodeRuntimeMBean** provides.

Table 8–22 SsuRemotePointCodeRuntimeMBean Status Attributes

Attribute	Description
getValue()	Specifies the address of the network entity in the URI format where with the colon character (:) is replaced with the underscore.
getStatus()	Specifies a point code status. Possible values: ■ 0 - Remote SCCP is unavailable ■ 1 - Remote SCCP is available

Table 8–23 describes the attributes that **SsuSigtranM3uaRuntimeMBean** provides.

Table 8–23 SsuSigtranM3uaRuntimeMBean status Attributes

Attribute	Description
getAssociationName()	Specifies the name of the association.

Table 8–23 (Cont.) SsuSigtranM3uaRuntimeMBean status Attributes

Attribute	Description
getM3uaStatus()	Specifies the M3ua Status. Possible values: ■ 0 - down ■ 1 - inactive ■ 2 - active ■ 3 - down sent ■ 4 - up sent ■ 5 - active sent ■ 6 - inactive sent

Table 8–24 describes the attributes that **SsuRemoteSystemRuntimeMBean** provides.

Table 8–24 SsuRemoteSystemRuntimeMBean status Attributes

Attribute	Description
getValue()	Specifies the remote SCTP Connection.
getStatus()	Specifies the status of the remote SCTP Connection. Possible values: ■ 0 - lost or down ■ 1 - established or restarted

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Checking the Status of SMPP Network Entities

Using **SmppAdapterMBean**, you can get the status of all SMPP connections. Table 8–25 describes the attribute that **SmppAdapterMBean** provides.

Table 8–25 SmppAdapterMBean Status Attribute

Attribute	Description
getlistSmscConnectionStatus[ ]	Specifies the status of all SMSC connections. Possible values: ■ 0 - Inactive ■ 1 - Active

Using **NetworkEntityRuntimeMBean**, you can get the status of the network entity. Table 8–26 describes the attributes that **NetworkEntityRuntimeMBean** provides.

Table 8–26 NetworkEntityRuntimeMBean Status Attributes

Attribute	Description
getValue()	Specifies the address of the network entity in the URI format where with the colon character (:) is replaced with the underscore.

Table 8–26 (Cont.) NetworkEntityRuntimeMBean Status Attributes

Attribute	Description
getStatus()	Specifies a network entity status. Possible values: ■ 0 - Network entity is unavailable ■ 1 - Network entity is available ■ 2 - Status of the network entity is unknown

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Checking the Status of SIP Network Entities

Using **NetworkEntityRuntimeMBean**, you can get the status of the network entity. Table 8–27 describes attributes that NetworkEntityRuntimeMBean provides.

Table 8–27 SIP SSU Monitoring Attributes

Attribute	Description
getValue()	The attribute contains the address of the network entity in the URI format where with the colon character (:) is replaced with the underscore.
getStatus()	Specifies a network entity status: ■ 0 - Network entity is unavailable ■ 1 - Network entity is available ■ 2 - Status of the network entity is unknown

For more information on how to access runtime MBeans, see the discussion on monitoring Service Controller using runtime MBeans in *Service Controller System Administrator's Guide*.

Checking Availability of the SIP Port

To ensure a stable communication between entities in a SIP network and Service Controller, you can check whether the UDP SIP port of Service Controller is up. This check is known as healthcheck.

To perform a healthcheck for a UDP SIP port:

- Configure the load balancer to send to Service Controller a raw UDP package with the following data bytes (hex): 70 69 6e 67.

  Service Controller recognizes this package as a healthcheck and does not parse the package.

9 Implementing Overload Protection

This chapter explains how to protect Oracle Communications Service Controller from overload. For more information, see the *Service Controller System Administrator's Guide*.

Implementing Overload Protection

In some cases, such as unanticipated traffic peaks or failure of a network hardware or software component, the load on Service Controller modules can increase significantly. This can cause a situation known as system overload in which Service Controller modules have insufficient resources to handle new sessions. If overload is not handled correctly, the system can crash and critical data can be lost.

To handle increased amounts of traffic without damaging operations of the entire system, Service Controller provides an overload protection mechanism. This mechanism operates in Processing Domains where you can define criteria for overload detection.

By default, an overload condition is triggered by an excessive number of 1) active sessions or 2) initial requests. The system rejects initial requests while the overload lasts. In addition to rejecting initial requests, Service Controller provides the capability to customize how the system behaves if overload occurs.

Using Gauges and Counters as Key Overload Indicators

When you configure gauges and counters as key overload indicators, Service Controller triggers overload protection if threshold values are crossed as measured by those indicators. You can select any of the counters and gauges provided by Service Controller to serve as key overload indicators.

Note: Consult with Oracle Technical Support if you have questions about which runtime MBeans are best suited to implement overload protection in your network environment.

When you configure overload protection, your settings are applied uniformly across all managed servers in the domain. Usually, server load balancing allocates traffic "fairly" across servers. However, it is possible for a single managed server in a domain to enter an overload condition while the other servers are functioning normally.

Understanding System and Module Levels of Overload Protection

Service Controller overload protection can be configured at the system and module levels:

- System counters and gauges: These two indicators can detect and trigger an overload condition that might occur across any number of clustered managed servers:
  - **`SystemCountersRuntimeMBean.SessionGauge`**
  - **`SystemCountersRuntimeMBean.InitialRequestCount`**

  The SessionGauge gauge represents the total number of active sessions on a single managed server (JVM).

  The InitialRequestCount counter represents the session creation rate. For example, the number of new sessions created per second on a single managed server.

  You configure the SessionGauge and InitialRequestCount indicators for the managed servers in the Administration Console. All managed servers share those configuration settings.

  There is no global overload protection status. However, if any managed server goes into an overload state, as defined by the shared configuration, the system stops accepting new sessions until the overload condition ceases.
- Module-level counters and gauges: These indicators are for specific modules. Using the Administration Console, you can configure these indicators for overload protection under the monitoring tabs. Expand Platform, then OCSB, then Processing Tier, and then Interworking and Supplementary Modules.

Understanding the Essential Steps for Configuring Overload Protection

These steps must be followed to configure overload protection.

1. By default, the following gauge and counter are defined as your key overload indicators:
   - **`SystemCountersRuntimeMBean.SessionGauge`**: A gauge that measures the number of active sessions handled by a single managed server. The number of active sessions includes all Service Controller sessions.
   - **`SystemCountersRuntimeMBean.InitialRequestCount`**: A counter that measures the rate at which a managed server receives new sessions.

     The SessionGauge and InitialRequestCount measurements are per managed server but all of the managed servers share a common configuration. If any managed server goes into an overload condition, the system stops accepting new sessions until the overload condition ceases.
2. Identify module-level counters and gauges you want to use as key overload indicators.
3. Configure Threshold Crossed Notifications details for your module-level counters and gauges.

   For each counter and gauge you want to use as a key overload indicator, define an upper threshold and ceased value threshold. For example, if the upper threshold value is 100 and the ceased value is 90 then if 100 is crossed the system remains overloaded until the value goes below 90.

   Specify a threshold name for each module-level counter or gauge. If you use either sessionGauge or initialRequestCount as the threshold name value you do not have to add these indicators to the Key Overload Indicators pane.

   See the *System Administrator's Guide* for more information about configuring runtime MBeans thresholds.

4. Configure Key Overload Indicators.

   After you have configured the module-level counters and gauges you want to use for overload protection you need to specify that they are to be used by Service Controller as Key Overload Indicators.

   You do this by using the Administration Console. Expand Tier Management, then use the Key Overload pane to configure your key overload indicators.

   Important: Service Controller activates overload protection when any of your key overload indicator crosses its upper threshold.

5. Customize overload protection behavior.

   The behavior of Service Controller is that if an overload condition occurs, the system continues to handle all active sessions but rejects initial requests until the overload condition ceases.

   In addition to the default protection behavior, you can customize how Service Controller responds to SIP and Diameter network entities that attempt to establish sessions during a system overload.

   Example: You can define the type of error and value of the SIP Retry-After header field that Service Controller uses to respond to newly established SIP sessions.

   You can customize overload protection behavior by using the Administration Console. Expand Tier Management, then Overload and Tracing, and then the Overload Protection Methods pane.

Configuring Key Overload Indicators

The following sections describe in detail how to configure Key Overload Indicators.

Configuring Threshold Crossed Notifications Rules

This section describes how to create Threshold Crossed Notifications rules for overload protection. The components of these rules specify MBean type, threshold name, crossed and ceased threshold values, and other fields.

The following steps are applicable to both system-wide and module-level counters and gauges. There are only two system-wide overload indicators: **`sessionGauge`** and **`InitialRequestCount`**. The default settings for these indicators should usually not be changed.

To transform the counter or gauge you configure in this section to be a key overload indicator you must match the threshold name value you set under the Monitoring tab with the threshold name value in the Key Overload indicators pane.

For example, the default key overload indicator threshold name **`sessionGauge`** matches the threshold name value in the default threshold crossed notifications rule also sessionGauge.

To configure Threshold Crossed Notification Rules do the following:

1. In the navigation tree, expand the **OCSB** node.
2. Expand **Processing Tier**.
3. Do either of the following:
   - To configure System-level Counters and Gauges: Expand Tier Management, then Monitoring, and then Monitoring. **Note**: You cannot add more counters and gauges in addition to the default sessionGauge and InitialRequestCount

indicators. However, if required you can modify details such as crossed and ceased threshold values.

- To configure Module-level Counters and Gauges: Expand Interworking or Supplementary Modules, then Expand the module for which you want to configure a counter or gauge as an overload indicator.

4. In the Monitoring tab, select **Threshold Crossed Notifications**.
5. Be sure you have selected **Lock & Edit** and then click **New**.
6. In the **Threshold Name** field, enter a string that names the threshold. This value is referenced by the key overload indicators.
7. For the **Enable threshold** field, select **True** or **False**. Only enabled thresholds are considered for overload protection.
8. In the **MBean Type** field, enter the type of MBean.
9. For the **Counting Type** field, select the Counting method. For gauges use CurrentGeneralValue and for counters use CurrentIntervalDeltaValue.
10. In the **MBean Attribute** field, enter an MBean attribute. For SystemGaugeRuntime use SessionGauge and for SystemCountRuntime use InitialRequestCount.
11. In the **Threshold class** field, enter **High**. Crossing a low threshold does not cause an overload state.
12. In the **Threshold Value** field, enter an integer value which when crossed triggers an overload state.
13. In the **Threshold ceased value** field, enter an integer value which when crossed the triggered threshold ceases. This value is applicable only to gauges.
14. In the **Threshold crossed message** field, enter a message included in the threshold notification.
15. In the **Threshold ceased message** field, enter a message included in the threshold ceased notification.
16. In the **Server filter** field, leave it empty or use a regular expression to filter on a managed server. For example "managed_1" or "server."
17. In the **Resource filter** field, enter a unique name for the indicator.
18. Click **Apply**.

Specifying Your Key Overload Indicators

Identify the module-level counters and gauges you want to use for overload protection. Use the Administration Console to list the names and threshold names for these indicators.

The Overload Protection pane is pre-populated with these two system-wide indicators:

- **`SystemCountersRuntimeMBean.SessionGauge`**
- **`SystemCountersRuntimeMBean.InitialRequestCount`**

To specify module-level Key Overload Indicators do the following:

1. In the navigation tree, expand the **OCSB** node.
2. Expand **Processing Tier**.
3. Expand **Tier Management**.

4. Select **Overload Protection**. The **Key Overload Indicators** pane appears.
5. Be sure you have selected **Lock & Edit** and then click **New**.
6. In the **Name** field, enter a unique name for the indicator.
7. In the **Threshold Name** field, enter a unique string that references the threshold.

   Multiple indicators at the system or module-level can use the same Threshold Name. In this situation, all matching crossed thresholds will be considered to indicate an overload state.

   Example: If any module-level counter or gauge use either sessionGauge or initialRequestCount as the threshold name value, you do not have to add these module-level indicators to the Key Overload Indicators pane. However, the module-level settings (e.g. crossed threshold value) will override the platform-level settings for that individual module only.
8. Click **Apply**.

Configuring General Monitoring Parameters

This section describes how to configure general attributes for overload protection notifications.

These attributes can be configured both at the system and module levels. Expand Tier Management, then Monitoring, and then the General tab. To configure IMs, expand Interworking Modules, then the IM you want to configure, then Monitoring, and then the General tab.

Table 9–1 describes the configuration parameters on the Monitoring General tab. At the Tier level these parameters affect only SessionGauge and InitialRequestCount. At the module level the parameters affect all runtime MBeans in the module.

Table 9–1 General Overload Configuration Parameters

Name	Description
Enable runtime MBeans	Disables the runtime MBeans so you can neither poll them for values or get notifications.
Enable Notifications	Disables only notifications, so you can still poll values from the MBean.
Counter Interval (sec)	This parameter specifies the length of the interval in seconds. Note: This parameter is not configurative at the module level.
Notification trigger interval (sec)	Sampling interval in seconds for checking notifications. For example, if the Counter Interval is set to 10 seconds and the Notification trigger interval is set to 2 seconds for each counter interval the system will determine 5 times whether the threshold value has been crossed.

Configuring the Overload Protection Behavior

When system overload occurs, Service Controller rejects new sessions and sends response messages to the network entities that attempted to establish the new sessions.

In the Overload Protection Methods pane you can configure how Service Controller responds to attempts by SIP and Diameter network entities to establish new sessions.

Table 9–2 describes configuration parameters on the Overload Protection Methods subtab. Expand Platform, then Processing Tier, then Overload and Tracing, and then the Overload Protection Methods subtab.

Table 9–2 Overload Protection Methods

Name	Type	Description
Enabled	BOOL	Specifies whether or not the overload protection methods in this table are enabled. Possible values: ■ TRUE ■ FALSE
SIP Response Status Code	STRING	Specifies a SIP error that Service Controller returns to a SIP network entity when Service Controller declines an attempt to establish a session. Default value: 503
SIP Retry-After	STRING	Specifies the value that Service Controller sets in the Retry-After header of the error response sent to the network entity. This value defines how long the network entity waits before it retries to establish a session. Default value: 300
Diameter Response Result Code	STRING	Specifies a response Result Code AVP that Service Controller returns to a Diameter network entity when Service Controller declines the attempt to establish a session. Default value: 5012
Web Service Response Status Code	INT	Specifies an error code that Service Controller returns to a Web service network entity when Service Controller declines the attempt to establish a session. Default value: 503
SAL Response Status Code	INT	Specifies an error code that Service Controller returns to a SAL application when Service Controller declines the attempt to establish a session. Default value: 503

10 Implementing Keep Alive Session Timer

This chapter explains how to protect Oracle Communications Service Control by detecting 'dead/inactive' sessions earlier and purged the 'dead/inactive' sessions instead of waiting the max_session_duration. theoretically the feature is available for all signalling protocols e.g. diameter, ss7, sip. especially fit for long duration diameter sessions as customer requires to purge the 'dead/inactive' sessions.

Implementing Keep Alive Session Timer

An optional keepAliveSessionDuration configuration is added to implement the feature. The keep alive session timer is independent of maximum call duration timer. customer can optionally use both keep alive session timer and maximum call duration timer or only use one of them.

Note: the keep alive session timer or maximum call duration timer can be disabled by setting a zero or negative value (e.g. -1).

Configuring the Keep Alive Session Timer

The keep alive session timer is independent of maximum call duration timer. Customer can optionally use both keep alive session timer and maximum call duration timer or only use one of them. while we usually suggest customer to disable the max call duration timer then only utilize the keep alive session timer to do the housekeeping for long duration data services:

- set maxActiveSessionDuration = -1 to disable the max call duration timer in PN Framework and IM modules.

  take PN Framework for an example:

  http://${PN_Admin_IP}:${PN_Admin_Port}/axiaConsole

  Goto Configuration MBeans -> com.convergin.wcs.osgi.framework.pn.impl -> 6.2.0 -> maxActiveSessionDuration

  Set maxActiveSessionDurationz: -1   //disable the maxActiveSessionDuration timer. (ie. only use the keep alive session timer to do housekeeping)

- add the keepAliveSessionDuration configuration

  http://${PN_Admin_IP}:${PN_Admin_Port}/axiaConsole

  Goto Configuration MBeans -> com.convergin.wcs.osgi.framework.pn.impl -> 6.2.0 -> addKeepAliveSessionDuration

Click to execute Then you will see newly added mbean attribute "keepAliveSessionDuration"

Set value of keepAliveSessionDuration in seconds(e.g. 21600s, 6h)

The implementation has set an initial delay(20s) for the periodic keep alive session timer. So the predicted maximum 'delayed' inactive session detection is 20s + 2keepAliveSessionDuration. e.g. if customer set the keepAliveSessionDuration = 21600s. the inactive session maybe clean up in 20s+2*21600s = 43220s later.

11

Implementing SSU Sigtran High Availability

Oracle Communications Service Controller 6.2 includes an enhancement on SSU server's Sigtran High Availability. This enhancement ensures the usecase like HLR Query runs smoothly in case of a SSU server's Sigtran link is down while other Sigtran links can be selected to send out the HLR query (e.g. map ATI message) instead.

Implementing SSU Sigtran High Availability

The SSU server monitors the link status(AVAILABLE/UNAVAILABLE) and report it to PN server via two means:

- Immediate link Status report
- Periodical current link status report

The 'Immediate link Status report' triggers when one of the link status change. The SS7 stack detects the links status change at first and then SSU instance would trigger immediate link status report propagating to PN server via event broker. Meanwhile SSU instance could trigger 'Periodical current link status report' propagating to PN server(by default 30s). the current link status report includes all link status of a SSU instance.

The PN server processes the link status report. so that be able to track each remote destination status of each ss7 protocol adapter instance in the SSU server. when PN server need to send out the HLR query outbound event. The PN server start to select an AVAILABLE ss7 protocol instance for the remote destination in the round robin way. if one of the ss7 protocol instances become UNAVAILABLE then PN server would fail over to other AVAILABLE ss7 protocol instance. The PN server marks a remote destination UNAVAILABLE only if all ss7 protocol adapter instances in all SSU servers are 'disconnected' to this destination!