Oracle® Communications Service Broker

Orchestration User's Guide

Release 6.1

E29453-01

February 2013

ORACLE®

Oracle Communications Service Broker Orchestration User's Guide, Release 6.1

E29453-01

Contents

A Use Cases

B Initial Filter Criteria

Preface

This document provides a description of Oracle Communications Service Broker orchestration capabilities.

Audience

This document is intended for system administrators.

This document assumes that you are familiar with the following:

- Initial Filter Criteria (iFC)
- Session Initiation Protocol (SIP)
- IP Multimedia Subsystem (IMS) architecture and interfaces

Related Documents

The following documents provide additional information about Service Broker and orchestration logic.

- *Oracle Communications Service Broker Concepts Guide*
- *Oracle Communications Service Broker Installation Guide*
- (Optional) *Oracle Communications Service Broker Service Controller Implementation Guide*
- (Optional) *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*
- (Optional) *Oracle Communications Service Broker Social Voice Communicator Implementation Guide*
- (Optional) *Oracle Communications Service Broker VPN Implementation Guide*
- (Optional) *Oracle Communications Service Broker Subscriber Store User's Guide*
- *Oracle Communications Service Broker System Administrator's Guide*
- *Oracle Communications Service Broker Modules Configuration Guide*
- *Oracle Communications Service Broker Signaling Server Unit Configuration Guide*
- *Oracle Communications Service Broker Security Guide*
- (Optional) *Oracle Communications Service Controller Release Notes*
- (Optional) *Oracle Communications Online Mediation Controller Release Notes*

Downloading Oracle Communications Documentation

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

http://edelivery.oracle.com

Additional Oracle Communications documentation is available from Oracle Technology Network:

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

Documentation Accessibility

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

Access to Oracle Support

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

1

Application Orchestration Overview

This chapter provides an overview of application orchestration and describes how Oracle Communications Service Broker performs application orchestration.

About Application Orchestration

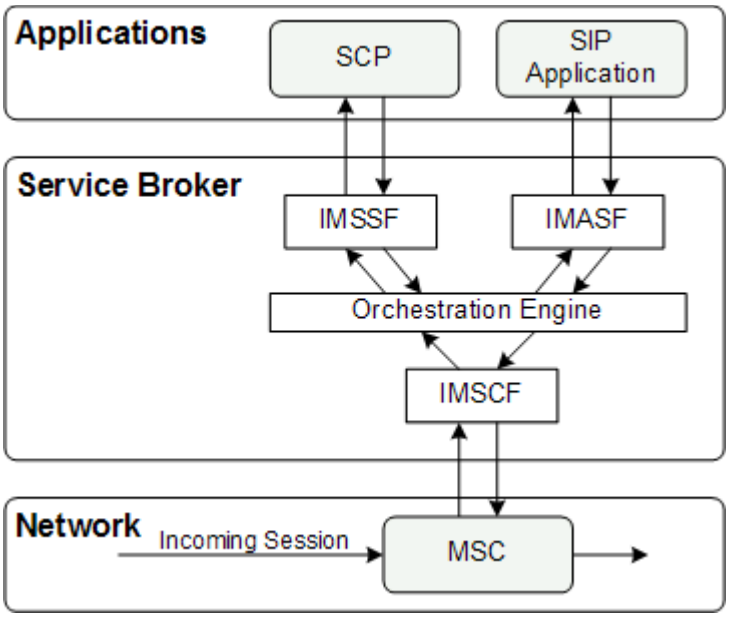
Orchestration is the ability of Service Broker to route a session through various applications. Service Broker routes a session sequentially, from one application to another. Each application executes a certain business logic. Every application applies a service on the session before Service Broker routes the session to a next application.

Application orchestration is performed by the Orchestration Engine as follows:

1. A session arrives to the Orchestration Engine through a network-facing module.
2. The Orchestration Engine routes the session sequentially through various applications by using application-facing modules. You define the applications that the Orchestration Engine invokes, the order in which the Orchestration Engine invokes the applications, and conditions for invoking applications using a special notation known as the orchestration logic.
3. After the session passed all applications in the chain, the Orchestration Engine returns the session back to the session control entity in the network.

   Figure 1–1 shows an example of how the Orchestration Engine routes a session that arrives from the network through an SCP, then through a SIP application, and then back to the network.

Figure 1–1 Routing a Session Sequentially through Multiple Applications



About Orchestration Logic

Orchestration logic is a notation that you use to specify the applications that the OE invokes, the order in which the OE invokes these applications, and conditions for invoking the applications. You specify an orchestration logic for each subscriber.

The subscriber's orchestration logic is stored as a part of the subscriber's profile. Depending on your deployment of Service Broker, subscribers' profiles and orchestration logic can be defined in:

- Home Subscriber Server (HSS), which is the primary user database in the IMS domain. It contains subscription-related information including subscriber applications and subscriber profiles. The HSS OPR uses the Diameter protocol over the standard Sh interface to connect the HSS and select the subscriber profile.
- 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 OPR connects the LSS to look up subscriber profiles with the orchestration logic.
- Local Subscriber Store or BRM Subscriber Store, which is a database in which subscriber profile data is stored.
- Pre-defined list of applications that Service Broker should invoke.

Table 1–1 describes components that the OE uses to retrieve a subscriber profile from a profile server and execute the orchestration logic.

Table 1–1 Service Broker Components Responsible for Retrieving and Executing Orchestration Logic

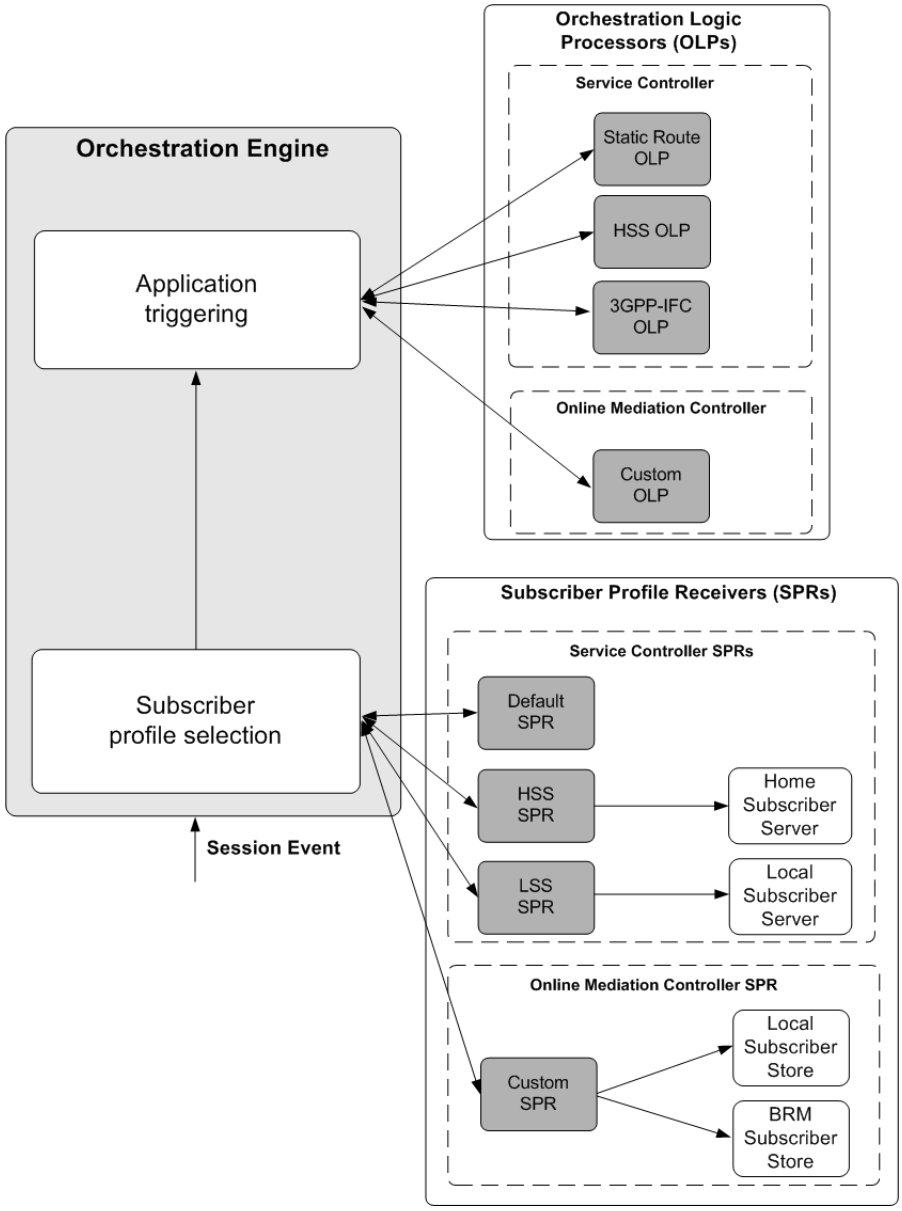
Component	Description
Subscriber Profile Receiver (SPR)	Connects to the profile server and retrieves the subscriber's profile with the orchestration logic. There are different types of SPRs to connect to different types of profile servers.

Table 1–1 (Cont.) Service Broker Components Responsible for Retrieving and Executing Orchestration Logic

Component	Description
Orchestration Logic Processor (OLP)	Retrieves the orchestration logic from a subscriber's profile and executes the orchestration logic.

Figure 1–2 shows how the OE retrieves subscriber profiles and executes the orchestration logic.

Figure 1–2 Retrieving Subscriber Profiles and Executing Orchestration Logic



When a new session arrives to the OE, the OE operates as follows:

1. The SPR connects to the profile server and retrieves the subscriber profile.
2. The OLP obtains the orchestration logic from the subscriber profile and triggers the applications as specified in the orchestration logic.

3. Then the OE releases the session.

About Subscriber Profile Receivers and Orchestration Logic Processors

The OE uses different SPRs to connect to different profile servers. When configuring the OE, you specify the appropriate SPR for the profile used in your system. Depending on the SPR you selected, you need to configure a corresponding Orchestration Logic Processor (OLP).

Table 1–2 explains which SPR you should select and which corresponding OLP you should configure depending on where the orchestration logic is defined.

Table 1–2 SPRs and Corresponding OLPs

To Execute the Orchestration Logic...	Select...	Then Configure...
Stored in a Home Subscriber Server (HSS)	HSS SPR	HSS OLP
Stored in a Local Subscriber Server (LSS)	LSS SPR	SM-LSS
Stored in a Local Subscriber Store or BRM Subscriber Store	Custom SPR	Custom OLP
Defined as a a pre configured list of applications	Default SPR	Static Route OLP

See the discussion on configuring the Orchestration Engine in *Oracle Communications Service Broker Modules Configuration Guide* for more information about specifying an OPR.

Note: You can add a new OPR to Service Broker, to connect to other profile sources that exist in the operator's network. Service Broker can apply orchestration logic defined in HSS or any other profile source to the legacy domain.

2

Configuring the Orchestration Engine

This chapter describes how to configure the Oracle Communications Service Broker Orchestration Engine.

Setting Up the Orchestration Engine

You set up the Orchestration Engine (OE) using the OE configuration screen.

To access the OE configuration screen:

1. In the domain navigation pane, expand **OCSB**.
2. Expand **Processing Tier**.
3. Select **Orchestration Engine**.

Table 2–1 describes the tabs available on the OE configuration screen.

Table 2–1 OE Configuration Subtabs

Task	Description
General	Enables you to specify a subscriber profile receiver and enable Service Data Records (SDRs) generation. See "Configuring General Parameters" for more information.
Static Route OLP	Enables you to specify applications that the OE should invoke and the order in which they are invoked. This tab is ignored if the OE is not configured to work with the Static Route orchestration logic processor (OLP). See "Configuring Static Route OLP Parameters" for more information.
HSS OLP	Enables you to set up the OE connection to an Home Subscriber Server (HSS). This tab is ignored if the OE is not configured to work with the HSS OLP. See "Configuring HSS OLP Parameters" for more information.
Custom OPR	Enables you to specify the name of the OPR that the OE should use to retrieve subscriber profiles. See "Configuring Custom OLP Parameters" for more information.
Monitoring	Enables you to define how logging and notifications operate. See "Configuring Monitoring Parameters" for more information.

Configuring General Parameters

The General subtab enables you specify a subscriber profile receiver (SPR) and enable SDR generation.

Table 2–2 describes configuration parameters on the **General** subtab.

Table 2–2 General Parameters

Name	Type	Description
Subscriber Profile Receiver	STRING	Specifies which SPR the OE uses to retrieve an orchestration profile. Possible values: ■ OlpDefaultInfoReceiver Select this option when you want the OE to use the static route OLP. To define the static route, use the Static Route OLP tab. See "Configuring Static Route OLP Parameters" for more information. ■ OlpLSSInfoReceiver Select this option when you want the OE to retrieve subscriber profiles from an SM-LSS. See the discussion on configuring an SM-LSS in *Oracle Communications Service Broker Modules Configuration Guide*. ■ OlpHSSInfoReceiver Select this option when you want the OE to retrieve subscriber profiles from an HSS. To define the address of the HSS, use the HSS OLP tab. See "Configuring HSS OLP Parameters" for more information. ■ OlpCustomInfoReceiver This option is relevant for the Online Mediation Controller only. Select this option when you want the OE to retrieve subscriber profiles from the Subscriber Store. You can enable the Degraded Mode only when this option is selected. See "Configuring Custom OLP Parameters" for more information.
Enable SDR	BOOL	Specifies whether or not the OE generates SDRs. Possible values: ■ True ■ False Default value: True

Table 2–2 (Cont.) General Parameters

Name	Type	Description
Enable Session Persistency	STRING	Specifies the point in a call when session persistency begins. Persistency continues throughout the session with each new state overwriting the previous state in the repository. ■ When Session Starts Persistency begins when the first session setup message is received. The current state of the session is then stored in the persistent repository. Each state is overwritten by the state that follows it until the end of the session. ■ On Ringback Persistency begins when a ringing indication is received. The current state of the session is then stored in the persistent repository. Each state is overwritten by the state that follows it until the end of the session. ■ On Answer Persistency begins when an answer indication is received. The current state of the session is then stored in the persistent repository. Each state is overwritten by the state that follows it until the end of the session. ■ Never No state of the active session is stored.

Configuring Static Route OLP Parameters

The Static Route OLP subtab enables you to specify applications that the OE invokes and the order in which they are invoked.

Note: This tab is regarded only when the OE is configured to work with the Static Route OLP. In this case the Subscriber Profile Receiver parameter in the General tab is set to OlpDefaultInfoReceiver.

Table 2–3 describes the configuration parameter on the Static Route OLP subtab.

Table 2–3 Static Route OLP Parameter

Name	Type	Description

Default Routing Targets	STRING _LIST	Specifies a list of application SIP URIs that the OE must invoke.
		The format of a SIP URI is:
		module-instance-name.module-type@convergin.com
		You can specify several SIP URIs separated by a space.
		For example:
		`sip:IMSCFCAP4_instance.IMSCFCAP4@convergin.com`
		`sip:IMASF_instance.IMASF@convergin.com`

Configuring HSS OLP Parameters

In the HSS OLP tab you can define the address of the HSS that the OE connects, and you can optionally specify mobile subscribers for whom the OE obtains orchestration logic (iFCs) from the HSS.

> **Note:** This tab is regarded only when the OE is configured to work with the HSS OLP. In this case the Subscriber Profile Receiver parameter in the General tab is set to OlpHSSInfoReceiver.

Table 2–4 describes the configuration parameters on the HSS OLP tab.

Table 2–4 HSS OLP Parameters

Name	Type	Description
Wildcarded PSI	STRING	Specifies a regular expression that the HSS uses to search for a subscriber's orchestration logic (iFCs). The HSS compares the regular expression against Public Subscriber Identities (PSIs) in its database. The HSS finds all matches and respond to the OE with one or more iFCs that comprise the subscribers orchestration logic. You need to specify a regular expression in a SIP URI format. You can use the following wildcards: ▪ asterisk (*), which matches zero or more occurrences of any character. For example, sip:78880*@oracle.com matches sip:78880@oracle.com and sip:788801@oracle.com. ▪ period (.), which matches one occurrence of any character. For example, sip:78880.0@oracle.com matches sip:7888010@oracle.com and sip:7888020@oracle.com. ▪ exclamation mark (!), which represents any number of characters in the middle of the PSI or at the end of the PSI. For example, sip:78880!@oracle.com matches sip:subscriber10@oracle.com and sip:subscriber11@oracle.com If you specify this parameter, it prevails the session headers, and session headers are ignored. Leave the parameter empty to have the HSS search an orchestration logic for a subscriber, based on the **To** and **From** headers of a session. It is recommended to use this parameter when a group of subscribers share the same orchestration logic.
Destination-Host AVP	STRING	Specifies the host name of the destination HSS. The OE sets this value in the Destination-Host AVP, inside the UDR that it sends to the HSS. Note that this value must correlate to either a PeerMBean or a RouteMBean that you already configured in the Diameter SSU.
Destination-Realm AVP	STRING	Specifies the value that the OE sets in the Destination-Realm AVP, inside the UDR that it sends to the HSS.

Configuring Custom OLP Parameters

In the Custom OLP tab, you specify the name of the OPR that the OE should use to retrieve subscriber profiles. This is relevant for Online Mediation Controller only. Do not change the name of the OPR.

If the subscriber profile is missing, the OPR returns the error, and the OE terminates the session.

Configuring Monitoring Parameters

The Monitoring tab enables you to define how Runtime MBeans and notifications operate for the OE. For more information about configuring monitoring, see the discussion on configuring Service Broker monitoring in *Oracle Communications Service Broker System Administrator's Guide*.

Routing a Session through Non-Configured Applications

Typically, all applications in a production system are known. In this case, you define an individual IM-ASF module instance to communicate with each application. In this case, orchestration logic (for example, iFC) turns a session through various applications through different IM-ASF module instances.

There are cases in which the Orchestration Engine is required to orchestrate each session differently, each through a different application. In this case, it is impossible to pre-configure the different application addresses, either because there are many of them or their address is subject to change. The application addresses are not know to Service Broker.

To support orchestration with non-configured applications, you need to define a special instance of an IM-ASF module known as default IM-ASF. This instance will not be limited to interaction with only a single pre-configured application, but will rather allow interaction with any application. This instance must be named "IMASF_default".

Whenever the Orchestration Engine is required to route a session to a non-configured application, it will route it through "IMASF_default" module. When triggered, "IMASF_default" forwards a session to any application, as specified inside the session request, in the application address field.

For example, if the Orchestration Engine has to route a session to a non-configured application address, such as "sip:209.95.109.191:5060", the Orchestration Engine forwards this session to the default IM-ASF. The default IM-ASF forwards the session to the application server which IP address is 209.95.109.191.

For information on creating and configuring IM-ASF, see the discussion on setting up IM-ASF SIP in *Oracle Communications Service Broker Modules Configuration Guide*.

3

About the Orchestration Studio User Interface

This chapter describes the user interface of the Oracle Communications Service Broker Orchestration Studio.

About the Orchestration Studio

The Orchestration Studio provides an intuitive user interface that you use to graphically build an orchestration logic. With the Orchestration Studio, the process of building an orchestration logic consists of adding components of the logic to the Orchestration Studio canvas, specifying parameters of these components, and defining the flow between the components.

While creating a graphic representation of an orchestration logic flow in the Orchestration Studio, the Studio concurrently generates an XML-based file in the Initial Filter Criteria (iFC) format.

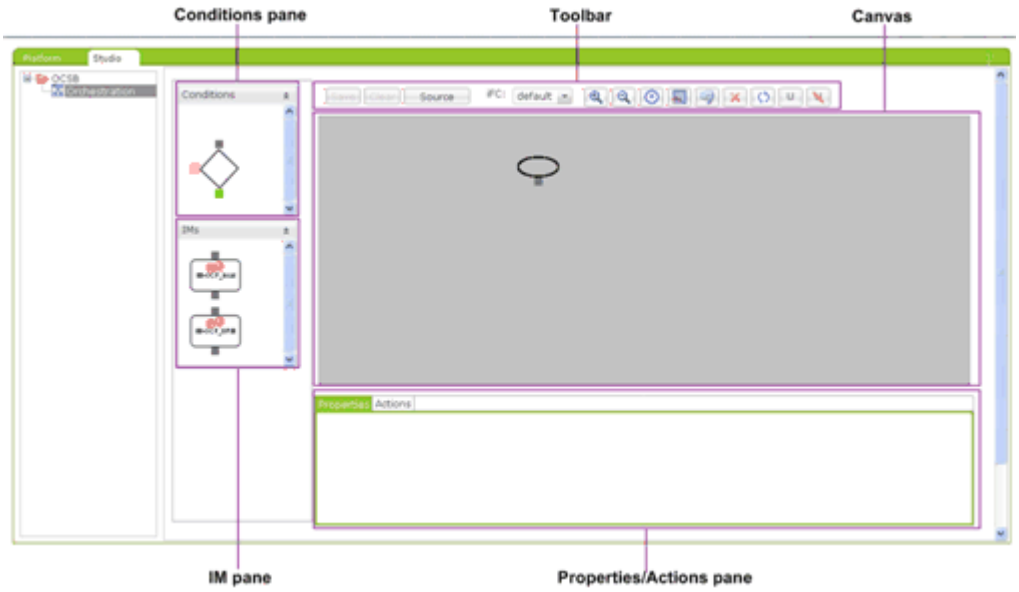
The orchestration logic that you build with the Orchestration Studio is saved in SM-LSS.

Note: The Orchestration Studio is not supported by Microsoft Internet Explorer.

Around the Orchestration Studio User Interface

Figure 3–1 shows the Orchestration Studio interface.

Figure 3–1 Orchestration Studio Interface



Conditions Pane

The Conditions pane contains the icon that you use to set conditions for routing a session. See "Building a Conditional Orchestration Logic Flow" for more information.

IM Pane

The IM pane contains the icons that represent IMs configured in your Service Broker deployment. You use these icons to define the IMs to which the OE routes the session. See "Specifying IMs" for more information.

Canvas

The main pane is the canvas where you create orchestration logic flows.

Note: The **Start** icon is displayed in the canvas by default and cannot be removed.

Table 3–1 describes the items in the area above the canvas:

Table 3–1 Orchestration Studio Menu Bar

Name	Type	Use
Save	Button	Saves the active orchestration logic flow locally.
Clear	Button	Removes all icons and their settings from the canvas.
Source	Button	Displays the iFC representation of the orchestration logic that you created using the Orchestration Studio.
iFC	Drop-down list	Contains the names of existing orchestration logic flows stored in SM-LSS.

Properties / Actions Pane

The pane beneath the canvas contains two tabs: **Properties** and **Actions**.

Properties

The Properties tab displays the configuration of the selected IM in the canvas. A selected element is outlined in red.

Only IMs that were configured for your Service Broker deployment are available in Orchestration Studio.

You can modify the configuration parameters in the **Properties** tab view and the updated configuration is reflected in the **Platform** tab.

Actions

The **Actions** tab contains the actions that can be applied to the selected **Conditions** icon or to the selected IM icon in the canvas. A selected icon is outlined in red.

Table 3–2 describes the options in the applications **Actions** tab.

Table 3–2 Orchestration Studio Actions Tab: IMs

Name	Use
Server Name	Alias of the IM to which the OE routes the session.
Default Handling	Determines if a session continues when the application returns an error or is unavailable.
Unidirectional Group	Forces the OE to route messages sent by the called party in the same direction as messages sent by the calling party.
Back to Back	Forces the OE to continue the session when an application sends a 3xx response code.
Response	Specifies the response code which forces the OE to continue the session. This field is available only if Back to Back is selected.
Tag	Tags the applications through which the OE routes the session.

Table 3–3 describes the buttons in the conditions on the **Actions** tab.

Table 3–3 Orchestration Studio Actions Tab: Conditions

Name	Use
New Group	Defines the group to which a condition belongs.
Delete	Removes the selected item.
SIP Request	SIP Request specifies the URI value of the destination application of server.
Session Description	Provides information about the type of communication between two end parties.
SIP Method	Specifies the SIP method name.
SIP Header	Specifies the name of a header that you want to check and the contents of the header. You can specify any standard SIP header as well as any custom header.
Tag	Allows you to create a condition that matches the tags applied to applications through which a session was routed.

You can drag the top of the Properties / Actions pane to increase or decrease its height.

Toolbar

Table 3–4 shows the buttons in the toolbar.

Table 3–4 Orchestration Studio Toolbar

Buttons	Name	Use
	Zoom In	Zooms in to different areas of the flow up to a magnification of 200%
	Zoom Out	Zooms out to 0.25% of the flow's actual size
	Resets Scale	Resets the canvas to 100%
	Select All	Selects all the icons in the flow. Selected icons are outlined in red.
	Undo	Reverts to the previous state
	Delete Selected	Deletes all selected icons from the flow
	Refresh	Refreshes the page
	Unidirectional Group	Forms two or more applications into a unidirectional group
	Cancel Unidirectional Group	Ungroups a unidirectional group

About Components of an Orchestration Logic Flow

To build an orchestration logic flow in the Orchestration Studio, you use the following components:

- **Condition** icon, which represents the conditions that determine how sessions are transferred to a specified application. Any flow must start with the Condition icon connected to the Start icon.

  You can connect the Conditions icon to other icons by connecting its connection points with the connection points of other icons. The Conditions icon has the following connection points:

  - **Input**, which is colored in grey. You use this connection point to connect the Start icon or IM icons with the Conditions icon.
  - **Yes**, which is colored in green. You use this connection point to connect the Conditions icon with an IM to which the OE routes the session when the conditions are met.
  - **No**, which is colored in red. You use this connection point to connect the Conditions icon with an IM to which the OE routes the session when the conditions are not met.
- **IM** icon, which represents the IMs in your Service Broker deployment. You can connect the IM icon to other icons by connecting its connection points with the connection points of other icons. The IM icon has the following connection points:
  - **Input**, which is located on the top of the icon. You connect a Yes or No connection point of a Condition icon with an IM icon using the Input connection point of an IM icon.
  - **Output**, which is located on the bottom of the icon. You connect an IM icon with an Input connection point of a Condition icon using the Output connection point of an IM icon.
- **Connector line**, which connects the icons at their connection points.

4

Building an Orchestration Logic Flow

This chapter describes how to use the Oracle Communications Service Broker Orchestration Studio to create a basic orchestration logic flow.

About Building Orchestration Logic with the Orchestration Studio

The process of building an orchestration logic flow with the Orchestration Studio consists of adding components of the flow to the canvas, specifying parameters of these components, and defining the flow between the components. The components being configured and connected to each other, comprise an orchestration logic flow. See "About Components of an Orchestration Logic Flow" for more information about components of an orchestration logic flow.

Using the Orchestration Studio, you can build the following types of orchestration logic flows:

- Conditional orchestration logic flows, in which the OE routes the session to different IMs based on the conditions that you set
- Unconditional orchestration logic flows, in which the OE routes the session to the next application without checking any conditions

Depending on whether you want to build a conditional or unconditional orchestration logic flow, you need to add and configure different Orchestration Studio components.

About Building a Conditional Orchestration Logic Flow

In a conditional orchestration logic flow, the OE routes the session to different IMs based on the conditions that you set. A condition checks whether a specific parameter in an incoming message is set to a specified value. If a condition is met, then the OE routes the message to the specified component of Service Broker. The orchestration logic can use parameters set both in headers and different parts of the message's body.

For example, you can specify that the OE invokes a bill shock prevent application only if the From header of the message contains a specific subscriber. Similarly, you can define the application to which the OE routes a session based on the subscriber's credit level stored in the subscriber's profile.

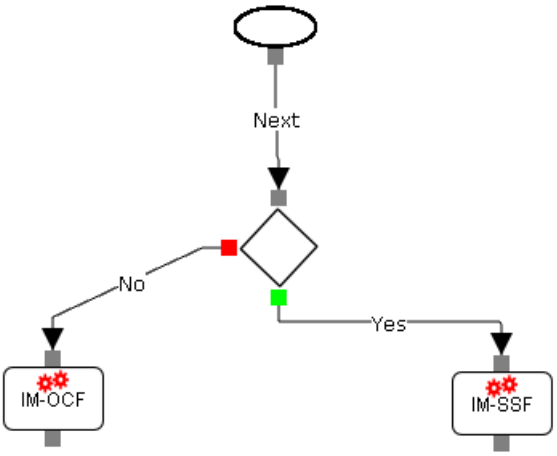
When building a conditional orchestration logic flow, you use the following components of an orchestration logic flow:

- Conditions, which define parameters whose values must be checked
- IM, to which the OE routes the session when the conditions are met
- IM, to which the OE routes the session when the conditions are not met

- Connector lines, which connect conditions and IMs

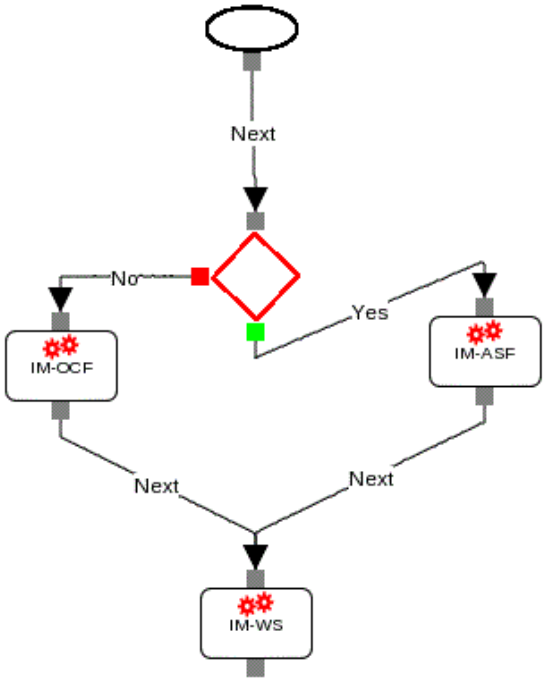
Figure 4–1 shows an orchestration logic where the session continues to the IM-SSF (see the Yes conditional branch in the figure), if the conditions are met and to the IM-OCF (see the No conditional branch in the figure), if the conditions are not met.

Figure 4–1 Conditional Orchestration Logic Flow



You can merge conditional branches at any point in the orchestration logic. Figure 4–2 shows the orchestration logic that routes the session to an IM-ASF if the condition is met (see the Yes conditional branch) or to an IM-OCF if the condition is not met (see the No conditional branch). After the session passed the IM-ASF or IM-ASF, the OE routes the session to IM-WS.

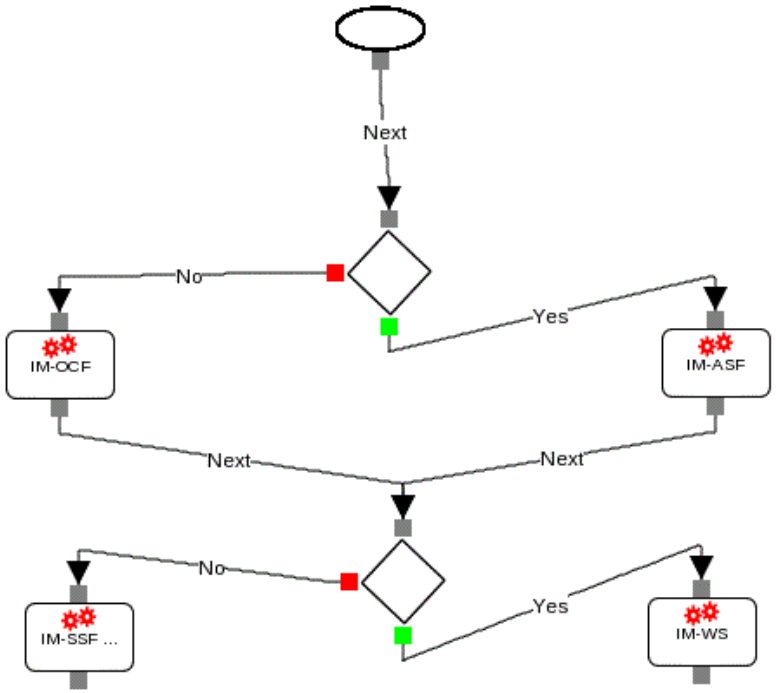
Figure 4–2 Merging Conditional Branches into an IM



Similarly, you can merge conditional branches into another condition. In this case, the OE checks whether the session that came through any of the merged routes meets the

specified conditions. Figure 4–3 shows the orchestration logic that performs an additional conditional check after the session passed IM-OSF or IM-ASF.

Figure 4–3 Merging Conditional Branches into a Condition



About Conditional Parameters

You can check parameters in the headers and in the body of an incoming message.

About Checking Conditional Parameters in the Headers of an Incoming Message

You can check the following parameters in headers of an incoming message:

- URI of the destination server
- SIP method of the incoming message
- Values of the specified SIP headers

About Checking Conditional Parameters in the Body of an Incoming Message

The message's body can consist of multiple parts. Each part carries a specific type of information. For example, one part of the message body can contain an XML representation of a CAP message, and another part can include a subscriber's profile. In addition, applications can add their own data to the message body.

In each part, data can be presented in a different format and be structured differently. For example, the structure of a CAP message differs from the structure of a Diameter message.

You can check parameters in any part of the message's body by querying the message body with an XPath expression. For example, if you want to query the message's body to check whether the subscriber's account defined in the subscriber's profile is active, you can construct the following XPath expression:
`/UserProfile/GlobalProfileData[AccountState=Active]`.

To query the message's body, you need to know the structure of the body type that you want to query. The structure of different body types is defined in XSD files. These files are provided with Service Broker. Table 4–1 explains where the XSD files are located.

Table 4–1 Location of the XSD Files Provided with Service Broker

Body Data	Structure Defined In...
SS7 messages	Protocol XSD files. These files are stored in the **/protocols directory** in **/samples/service_controller.samples.zip**.
Diameter messages	Diameter protocol XSD file. The file is stored in the **/ro directory** in **/samples/service_controller.samples.zip**.
Subscriber's Profile	**search.xsd**. This file is located in the bundle **oracle.ocsb.app.rcc.mediation.up2sal**.
Data added by applications	XSD files published by applications.

In addition, you need to define which part of the message's body you want to query by specifying the content type of this part. Notice that applications in the orchestration chain can add parameters to any part of the body. You need to specify the content type for the parameters added by applications.

About Limitations of XPath for Querying a Subscriber's Profile

When you query a subscriber's profile using XPath, you can use only a subset of the XPath standard. You can use only the following XPath operators:

- /
- []
- or
- and
- =, !=, <, <=, >, >=

About Grouping Conditions

You can build conditions of any complexity. Regardless of the complexity of a condition, you combine conditions into groups. After you added conditions to a group, you specify the logical operators AND or OR to be used between the conditional parameters of the same group and between different groups.

There are the following scenarios of how conditions can be grouped:

- Condition that consists of one conditional parameter

  For example, you can add the condition `SIP Method=INVITE`. In this condition, there is only one group. This group contains the conditional parameter SIP Method whose value is set to `INVITE`.

  See "Adding Conditions with One Conditional Parameter" for more information about adding these conditions.

- Condition that consists of multiple conditional parameters connected by AND or OR operators

  For example, you can define the condition `SIP Method=INVITE OR SIP Method=INFO OR SIP Method=SUBSCRIBE`. In this condition, there is only one

conditional group. The logical operator used between these conditional parameters is OR.

See "Adding Conditions with Multiple Conditional Parameters" for more information about adding these conditions.

- Condition that consists of multiple conditional expressions with each expression containing several conditional parameters. The conditional expressions are connected by AND or OR operators.

  For example, you can define the condition `(SIP Method=INVITE OR SIP Method=INFO) AND (From=John OR From=Paul)`. In this condition, there are two conditional groups:

  - The first group contains the conditional parameter `SIP Method` set to `INVITE` and another conditional parameter `SIP Method` set to `INFO`. The logical operator used between these conditional parameters is OR.
  - The second group contains the conditional parameter `From` set to `John` and another conditional parameter `From` set to `Paul`. The logical operator used between these conditional parameters is OR.

  The logical operator used between the groups is AND.

  See "Adding Conditions with Multiple Conditional Expressions" for more information about adding these conditions.

About Building a Unconditional Orchestration Logic Flow

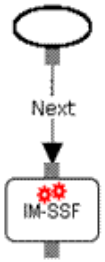
In an unconditional orchestration logic, the OE routes the session to the next application without checking any conditions.

When building an unconditional orchestration logic flow, you use the following components of an orchestration logic flow:

- IM, to which the OE routes the session
- Connector line, which connects IMs

Figure 4–4 shows the orchestration logic where the OE always routes the session to the IM-SSF.

Figure 4–4 Unconditional Orchestration Logic Flow



Building a Conditional Orchestration Logic Flow

In a conditional orchestration logic flow, the OE routes the session to different IMs based on the conditions that you set.

When you build a conditional orchestration logic flow, you perform the following actions:

1. Building a flow. See "Building a Flow" for more information.
2. Defining conditions. See "Adding Conditions" for more information.
3. Specifying the alias of the IMs that communicate with the applications to which you want the OE to route the session. See "Specifying IMs" for more information.

Building a Flow

To build a flow:

1. Do one of the following:
   - If you want to create an orchestration logic flow for an existing subscriber profile, go to the next step.
   - If you want to create an orchestration logic flow for a new subscriber profile, create a new subscriber profile using SM-LSS. For more information, see the discussion on configuring SM-LSS in *Oracle Communications Service Broker Modules Configuration Guide*.
2. Click the **Studio** tab.

   The Orchestration Studio configuration screen appears.
3. In the **iFC** list, select the subscriber profile for which you want to create an orchestration flow.
4. Drag-and-drop the **Condition** icon to the canvas.
5. Add conditional parameters. See "Adding Conditions" for more information.
6. Connect the **Start** icon with the input connection point of the condition icon.
7. Drag-and-drop the IM to which the OE routes the session if the conditions are met.
8. Connect the **Yes** connection point of the condition with the top connection point of the IM icon.
9. Drag-and-drop the IM to which the OE routes the session if the conditions are not met.
10. Connect the **No** connection point of the condition with the top connection point of the IM icon.
11. Specify aliases of both IMs as described in "Specifying IMs".
12. If required, add more conditions and connect them to the bottom connection point of IM icons.

Adding Conditions

The following sections describe how you can add conditions. See "About Grouping Conditions" for more information about conditions.

Adding Conditions with One Conditional Parameter

To add a condition with one conditional parameter:

1. On the canvas, select the condition icon.
2. In the **Properties/Actions** pane, click the **Actions** tab.
3. In the **Actions** tree, select the **Trigger Points** node.
4. Create a group by selecting the **Trigger Points** node and clicking **New Group**.

The Group subnode appears.

5. Select the newly created group.
6. Depending on the parameter that you want to check, click one of the following buttons:

Table 4–2 Conditional Parameters

You Want to Check...	Click...	Information to Be Provided
URI of the destination application server	**SIP Request**	In the **RequestURI** field, type the URI, using the following format: *sip:service@sip address:port number*.
Parameters in an SDP body	**Session Description**	In the **Line** field, type the one-character identifier of the parameter of you want to check. In the **Content** field, type the value of the parameter.
SIP method	**SIP Method**	From the **Method** list, select the method of the message.
SIP header	**SIP Header**	In the **Header** field, type the name of the header. In the **Content** field, type the value of the header.
Session tags	**Tag**	In the **Content** field, type comma-separated tags.
Parameters in the message's body	**Extended Session Description**	In the **Content Type** field, type the type of the body that you want to query. In the **XPath** field, type the XPath expression.

Adding Conditions with Multiple Conditional Parameters

To add a condition with multiple conditional parameters connected by AND or OR:

1. On the canvas, select the condition icon.
2. In the **Properties/Actions** pane, click the **Actions** tab.
3. In the **Actions** tree, select the **Trigger Points** node.
4. In the **Condition Type** list, select one of the following options:
   - If you want to add multiple conditional parameters connected by AND (for example, `SIP Method=INVITE AND SIP Header From=John`), select **OR set of ANDs**.
   - If you want to add multiple conditional parameters connected by OR (for example, `SIP Method=INVITE OR SIP Method=INFO`), select **AND set of ORs**.
5. Create a group by selecting the **Trigger Points** node and clicking **New Group**.
6. Select the node of the newly created group.
7. Depending on the parameter that you want to check, click one of the following buttons:

Table 4–3 Conditional Parameters

You Want to Check...	Click...	Information to Be Provided
URI of the destination application server	**SIP Request**	In the **RequestURI** field, type the URI, using the following format: *sip:service@sip address:port number*.
Parameters in an SDP body	**Session Description**	In the **Line** field, type the one-character identifier of the parameter of you want to check. In the **Content** field, type the value of the parameter.
SIP method	**SIP Method**	From the **Method** list, select the method of the message.
SIP header	**SIP Header**	In the **Header** field, type the name of the header. In the **Content** field, type the value of the header.
Session tags	**Tag**	In the **Content** field, type comma-separated tags.
Parameters in the message's body	**Extended Session Description**	In the **Content Type** field, type the type of the body that you want to query. In the **XPath** field, type the XPath expression.

8. Add as many conditions to the group as you need.

Adding Conditions with Multiple Conditional Expressions

To add a condition with multiple conditional expressions:

1. On the canvas, select the condition icon.
2. In the **Properties/Actions** pane, click the **Actions** tab.
3. In the **Actions** tree, select the **Trigger Points** node.
4. In the **Condition Type** list, select the option that determines the relationship between conditional group members and between groups (see "About Grouping Conditions" for more information about groups):
   - If you want to add multiple conditional expressions connected by OR with each expression consisting of conditional parameters connected by AND (for example, `(SIP Method=INVITE AND SIP Header From=John) OR (SIP Method=INVITE AND SIP Header From=Paul)`), select **OR set of ANDs**.
   - If you want to add multiple conditional expressions connected by AND with each expression consisting of conditional parameters connected by OR (for example, `(SIP Method=INVITE OR SIP Method=INFO) AND (SIP Header From=John OR SIP Header From=Paul)`), select **AND set of ORs**.
5. Create a group by selecting the **Trigger Points** node and clicking **New Group**.
6. Select the group node.
7. Depending on the parameter that you want to check, click one of the following buttons:

Table 4–4 Conditional Parameters

You Want to Check...	Click...	Information to Be Provided
URI of the destination application server	**SIP Request**	In the **RequestURI** field, type the URI, using the following format: *sip:service@sip address:port number*.
Parameters in an SDP body	**Session Description**	In the **Line** field, type the one-character identifier of the parameter of you want to check. In the **Content** field, type the value of the parameter.
SIP method	**SIP Method**	From the **Method** list, select the method of the message.
SIP header	**SIP Header**	In the **Header** field, type the name of the header. In the **Content** field, type the value of the header.
Session tags	**Tag**	In the **Content** field, type comma-separated tags.
Parameters in the message's body	**Extended Session Description**	In the **Content Type** field, type the type of the body that you want to query. In the **XPath** field, type the XPath expression.

8. Add as many conditions to the group as you need.
9. Create new groups and add conditions to these groups as required.

Specifying IMs

To specify the IM to which the OE routes the session:

1. On the canvas, select the IM icon.
2. In the **Properties/Actions** pane, click the **Properties** tab.
3. In the **Server Name** field, type the alias of the IM that communicates with the application to which the OE routes the session.

Building an Unconditional Orchestration Logic Flow

In an unconditional orchestration logic, the OE routes the session to the next application without checking any conditions.

When you build an unconditional orchestration logic flow, you perform the following actions:

1. Building a flow. See "Building a Flow" for more information.
2. Specifying the alias of the IM that communicates with the application to which you want the OE to route the session. See "Specifying IMs" for more information.

Building a Flow

To build a flow:

1. Do one of the following:

- If you want to create an orchestration flow for an existing subscriber profile, go to the next step.
- If you want to create an orchestration flow for a new subscriber profile, create a new subscriber profile using SM-LSS. For more information, see the discussion on configuring SM-LSS in *Oracle Communications Service Broker Modules Configuration Guide*.

2. Click the **Studio** tab.

   The Orchestration Studio configuration screen appears.
3. In the **iFC** list, select the subscriber profile for which you want to create an orchestration flow.
4. Drag-and-drop the IM icon to the canvas.
5. Connect the **Start** icon with the input connection point of the IM icon.
6. Specify the alias of the IM as described in "Specifying IMs".
7. If required, add more IMs to the orchestration chain.

Specifying IMs

The process of specifying IMs is the same as for the conditional orchestration logic flow. See "Specifying IMs" for more information.

Saving an Orchestration Logic Flow

You can save an orchestration logic flow at any time. Before saving, the Orchestration Studio checks whether the flow starts with the **Start** icon connected to the **Condition** icon that represents the first check in the flow.

If the **Start** icon is not connected to the **Condition** icon, the flow is invalid. The Orchestration Studio does not allow you to save an invalid flow. To save the flow, you need to connect the **Start** icon to the **Condition** icon.

To save an orchestration logic flow:

- Click **Save** on the menu bar.

  If the flow is valid, the Orchestration Studio saves it to SM-LSS. If the flow is not valid, the error message appears.

Removing an Orchestration Logic Flow or its Parts

You can remove any of the components:

- The entire orchestration logic flow
- Selected Conditions icons and IM icons from the flow
- Conditions from a group of Trigger Points

Removing an Orchestration Logic Flow

To remove an orchestration logic flow:

1. In the Orchestration Studio, select the name of the orchestration logic flow you want to remove from the **iFC** drop-down list.

   The saved flow is displayed.

2. Click **Clear** on the menu bar.

   Alternatively, on the tool bar, you can click **Select All** and then click **Delete Selected**.

3. Click **Save** on the menu bar and **Commit** from the banner.

   All icons and their settings in the flow are removed from the canvas.

   The orchestration logic file is removed from the database and from the profile name under which it was stored.

> **Note:** Removing an orchestration logic flow does not remove the subscriber profile. You remove a subscriber profile in SM-LSS. For more information, see the discussion on configuring SM-LSS in *Oracle Communications Service Broker Modules Configuration Guide*.

Removing Conditions and IMs from an Orchestration Logic Flow

To remove selected icons from the flow:

1. **Ctrl-click** the icons on the canvas you want to remove.
2. Click **Delete Selected** from the tool bar.

   The selected icons are removed from the flow.

3. Make sure you have not invalidated the flow by clicking **Save**.

   If you have invalidated the flow, a message appears on the screen explaining why.

Removing Conditions

To delete a group:

1. In the canvas, select the **Condition** icon containing the group you want to remove.
2. In the **Actions** tab, under Trigger Points, select the group you want to remove and click **Delete**.

   The group and all condition elements belonging to it are removed.

To delete individual conditions:

1. In the canvas, select the **Conditions** icon containing the SIP method or methods you want to remove.
2. In the **Actions** tab, under Trigger Points, select the SIP method under the Group number to which it belongs and then click **Delete**.

Removing Connector Lines from an Orchestration Logic Flow

To remove a connector line:

- Select a connector line in the canvas and click **Delete Selected** from the tool bar.

  Alternatively, you can press the **Delete** key.

  Only one connection line can be deleted at a time.

5

Providing Additional Information to an Application

This chapter describes how to use the Oracle Communications Service Broker Orchestration Studio to provide additional information to applications.

About Additional Information Provided to Applications

You can configure the OE to send additional information to an application. For example, you can inform the application that the calling party has been blacklisted for not meeting his billing obligations and that the OE should not complete the call.

You provide additional information to an application as a part of defining the IM that communicates with this application.

Providing Additional Information

To provide additional information to an application:

1. In the orchestration logic flow, select the **IM** icon of the IM that communicates with the application to which you want to provide additional information.
2. Click the **Actions** tab.
3. In the **Service Info** text field, type the information you want to provide to the application.

   The OE adds the text you type to the body of the message it forwards to the application.

6

Invoking Applications Based on the Previous Session Route

This chapter describes how to use the Oracle Communications Service Broker Orchestration Studio to set up an application to run based on the previous route of the session.

About Invoking Applications Based on the Previous Session Route

You can set up an application to run based on the previous route of the session. For example, you can set up a pre-paid application to run only if the session contains tags indicating that a home zone application precedes the pre-paid application in the orchestration chain.

To enable applications in the orchestration chain to get the information about applications through which the session is routed, you need to tag the session when it passes through an application. For example, if the OE routes the session to a home zone application, you can tag the session with the tag HomeZoneAppInvoked. This allows next applications in the orchestration chain to be aware of the hoe zone application was triggered.

Then you can set up one of the next applications in the chain (such as a bill shock prevention application) to run only if the session contains the HomeZoneAppInvoked tag.

As the session passes through applications in the orchestration chain, all tags are accumulated. Therefore, any subsequent application in the orchestration chain can be aware of those tagged applications which were previously triggered.

The OE adds tags to the session when routing it to the IM defined in the orchestration logic. You define a tag to be added as a part of the configuration of the appropriate IM.

Note: Service Broker does not support tags when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

Tagging a Session

To tag a session:

1. Select the **IM** icon of the IM that communicates with the application.

2. Click the **Actions** tab.
3. Click **Add**.
4. In the New dialog box, type the name of the tag you want to associate with the selected application and click **OK**.

   The tag name you defined is displayed in the **Tag** field. The Header field displays **x-wcs-tags**. This is the header that contains tags. You cannot modify the name of the header.

Checking Tags in a Session

You can set up an application to run depending on the application through which the session has already passed. For example, you can set up a condition that invokes the following applications in the chain:

- Home zone
- VPN
- Online charging

If, for example, the session does not contain a tag specifying it should access the VPN application, it skips the VPN application and continues to the pre-paid application.

You set up tags on a condition only when its icon is preceded by the **IM** icons representing the applications that added the specified tags to the session.

To set up tags on a condition:

1. Select the **Conditions** icon in the flow to which you want to apply a tag.
2. Click the **Actions** tab.
3. Select the group under which you want to apply the tag.
4. Click **Tag**.

   The **Header** field displays **x-wcs-tags**. This is the name of the header that contains tags added by the applications to the session. You cannot modify this text.
5. In the **Content** field, add the tag of the application to which you want the session to be routed.

   You can add as many tags as required, separated by a comma and a space.

   You can use regular expressions. For example, if your tag is **.*1234**.* the message is routed to all numbers that contain 1234, regardless of the numbers that precede or succeed it.

7

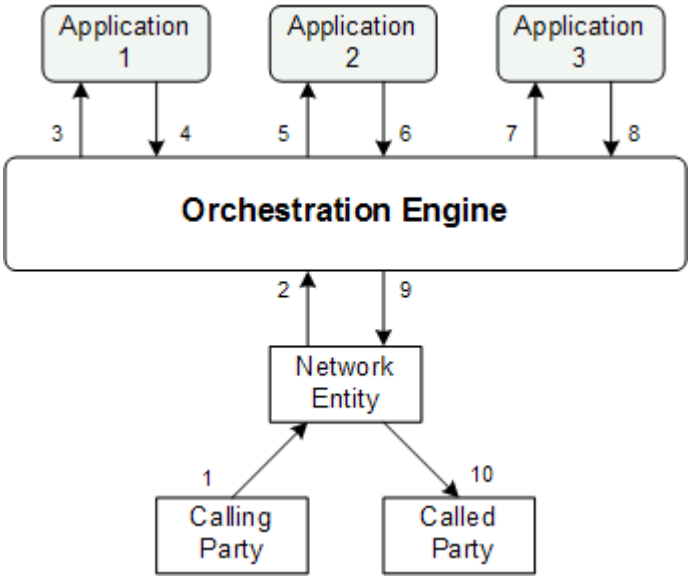
Defining the Orchestration Order of Messages Sent by a Called Party

This chapter describes how to use the Oracle Communications Service Broker Orchestration Studio to define the order of messages sent by a called party.

About the Orchestration Order

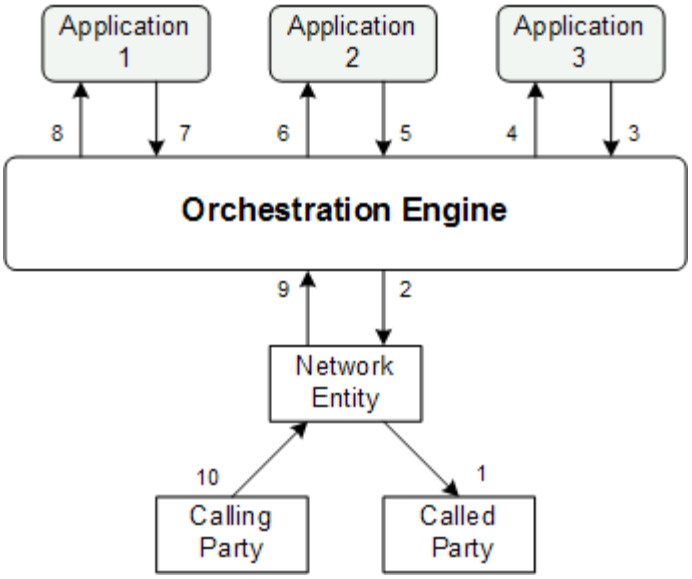
Orchestration logic defines how the OE routes messages generated by the calling party. For example, you can set an initial INVITE to be routed from Application 1 to Application 2 to Application 3. Figure 7–1 shows the order in which the OE routes an INVITE message from a calling party to a called party.

Figure 7–1 Routing an INVITE Message from a Calling Party



Orchestration logic does not specify how the OE routes messages received from a called party. By default, when a called party generates a message (for example, an OK response to an INVITE message), the OE routes this message in the reverse order, from Application 3 to Application 2 to Application 1. Figure 7–2 shows the order in which the OE routes an OK response from a called party to a calling party.

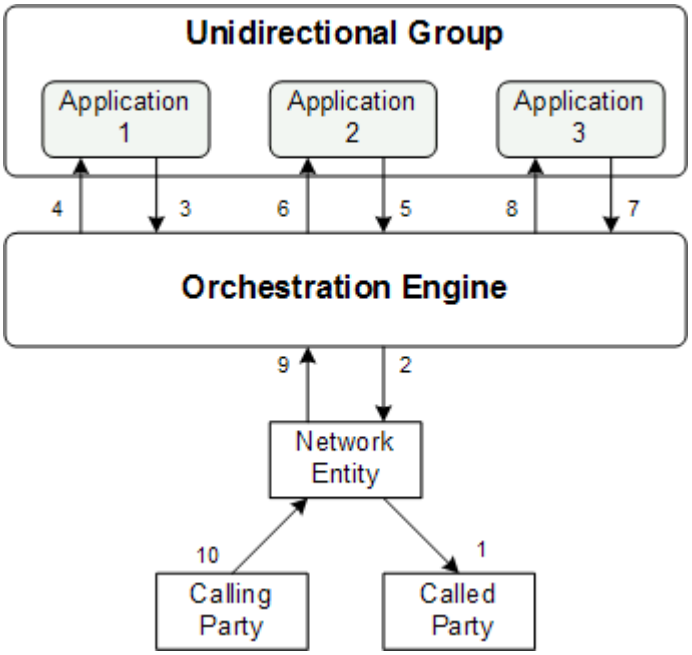
Figure 7–2 Routing an OK Message from a Called Party



When an application in the orchestration chain depends on the information generated by a previous application, you might need to route all messages, including those generated by a calling party and those generated by a called party, in the same order. For example, you might need a message to be first routed to an online charging application and then to a bill shock application. In this case, a bill shock application can perform certain actions based on the information generated by the online charging application.

To allow the OE to route all messages across applications in the same order, you need to group these applications in a unidirectional group. Figure 7–3 shows how the OE routes a message generated by a called party through applications in a unidirectional group.

Figure 7–3 Routing an OK Message from a Called Party in a Unidirectional Group



When grouping applications into unidirectional groups, you must observe the following limitations:

- You can group only those applications that run consecutively. For example, on Figure 7–3, you can group Application 1 and Application 2. However, you cannot group Application 1 and Application 3 because they do not run consecutively.
- Each application in a unidirectional group must be implemented as a Back-to-Back (B2B) application.

> **Note:** Service Broker does not support unidirectional groups when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

Grouping Applications into a Unidirectional Group

To group applications into a unidirectional group:

1. In the orchestration logic flow, press and hold the **Ctrl** key while selecting all **IM** icons of IMs that you want to form into a group.

   In the toolbar, the **Unidirectional** button becomes available.

2. Click the **Unidirectional** button.

   All the selected **IM** icons are now grouped.

   The Orchestration Studio automatically assigns an identifying numeral to the application group. Selecting any application within the group and clicking the **Actions** tab displays the unidirectional group number in the Unidirectional **Group** field. You cannot change the unidirectional group number in this field.

To ungroup a unidirectional group:

1. Select all the **IM** icons in the unidirectional chain.
2. Click the **Cancel Unidirectional Group** button.

8

Defining the Orchestration Engine Behavior on Receiving a Response from the Application

This chapter describes how to use the Oracle Communications Service Broker Orchestration Studio to define the Orchestration Engine (OE) behavior depending on the response that the OE received from the application.

About the Orchestration Engine Behavior on Receiving Responses from the Application

You can define whether the Orchestration Engine (OE) continues or terminates the session depending on the response that the OE received from the application. You can specify the following:

- Whether or not the OE continues the session when receiving an error from the application. See "Defining the Orchestration Engine Behavior on Receiving an Error from the Application" for more information.
- Whether or not the OE continues the session when receiving a response with the specified code from the application. See "Defining the Orchestration Engine Behavior on Receiving a Response from the Application" for more information.

Defining the Orchestration Engine Behavior on Receiving an Error from the Application

If an application returns an error, such as a **400 Bad Response** to an INVITE message, you can specify whether the OE forwards the message to the next application in the chain or terminates the session.

To specify the OE behavior:

1. In the orchestration logic flow, select the IM for which you want to specify default handling.

   By default, Default Handling is set to 0 to continue the session.

2. Click the **Actions** tab.
3. In the **Default Handling** field, type one of the following:
   - To continue the session, type **0**
   - To terminate the session, type **1**

Note: Service Broker does not support this feature when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

Defining the Orchestration Engine Behavior on Receiving a Response from the Application

You can configure the OE to forward the session to the next application whose conditions are met, when the OE receives a specific response from an application. The ability of the OE to forward the session to the next application is known as Forced Back to Back (FB2B).

To set Back to Back response codes:

1. In the orchestration logic flow, select the IM whose responses are to be forwarded Back to Back.
2. Click the **Actions** tab.
3. Select the **Back to Back** checkbox.

   The **Response** field becomes available.
4. In the **Response** field, type the response codes according to which the OE continues the session.

   You can add as many response codes as required, separated by a comma and a space.

See "Forcing Back to Back" for an example of a use case using FB2B.

Note: Service Broker does not support this feature when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

A

Use Cases

This appendix presents some typical use cases that can be used as examples when creating orchestration logic flows in the Oracle Communications Service Broker Orchestration Studio.

A.1 About the Use Cases

Some of the use cases in this appendix are based on the use cases described in *Oracle Communications Service Broker Concepts Guide*.

Using the procedures described in the earlier chapters in this guide to create orchestration logic flows, the use cases reproduced here demonstrate how you can build specific flows in the Orchestration Studio.

In addition, each use case displays the XML code that is generated by the Orchestration Studio while you graphically build the flow on the canvas.

The code contains some elements that are automatically inserted by the software and do not affect the behavior of the logic.

A.2 Service Orchestration

The following flows illustrate Service Broker orchestration capabilities.

A.2.1 IN Service Interaction

The following use case shows how the Orchestration Engine forwards a session to an online charging application server and then to a VPN service.

A.2.1.1 Building the IN Service Interaction Flow

To build an IN service interaction flow:

1. Drag the **Conditions** icon to canvas and draw a connecting line between **Start** and the icon.
2. Drag the **IM** icon representing IM-SSF INAP to the canvas and then drag the IM icon representing IM-SSF CAP to the canvas.
3. Draw a connector line from the green connection point on the **Conditions** icon to IM-SSF INAP and then draw a connector line from IM-SSF INAP to IM-SSF CAP.
4. Select the **Conditions** icon and in the **Actions** tab, click **New Group**.

   Group 0 is displayed under Trigger Point.
5. Select **Group 0**, click **SIP Method** and in the **Method** field, type INVITE.

6. Select the **IM** icon representing IM-SSF INAP and in the **Properties** tab, type the address of the destination online charging application server in the **Alias** field and click **Apply**.

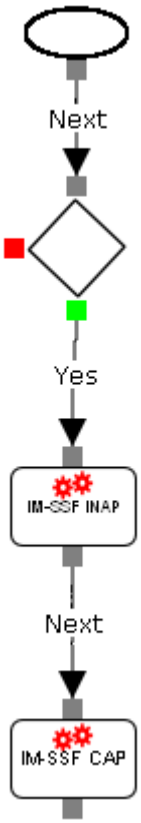
   The address of the application server must match the one set for the SCCP Local SSN.

   For more information, see the discussion on configuring IM-SSF in *Oracle Communications Service Broker Modules Configuration Guide*.

7. Select the IM icon representing IM-SSF CAP and in the Properties tab, type the name of the destination VPN application server in the Alias field and click **Apply**.

   Figure A–1 shows the orchestration logic flow for IN service interaction.

Figure A–1 IN Service Interaction Flow



A.2.1.2 Viewing the IN Service Interaction Source Code

Clicking **Source** displays the generated XML code of the flow you created graphically.

Figure A–2 shows the source code of the IN service interaction flow.

Figure A–2 IN Service Interaction Source Code

```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <Method>Invite</Method>
    </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IM-SSF INAP@ocsb</ServerName>
      <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
  <InitialFilterCriteria>
    <Priority>2</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <SIPHeader>
          <Header>x-wcs-history</Header>
          <Content>id=1.*</Content>
        </SIPHeader>
      </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IM-SSF CAP@ocsb</ServerName>
      <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

A.2.2 IMS Service Interaction

The following use case shows how Service Broker communicates with the IMS network and provides service interaction based on the logic retrieved from the database.

A.2.2.1 Building the IMS Service Interaction Flow

To build an IMS service interaction flow:

1. Drag the **Conditions** icon to the canvas and draw a connecting line between **Start** and the icon.
2. Drag the **IM** icons representing the following IMs to the canvas in the order in which they are listed:
   - IM-SSF

- IM-ASF
- IM-OCF

3. Draw a connector line from the green connection point on the **Conditions** icon to the IM-SSF.
4. Draw connector lines from the IM-SSF to connect the remaining **IM** icons in the order you placed them on the canvas.
5. Select the **Conditions** icon and in the **Actions** tab, click **New Group**.

   Group 0 is displayed under Trigger Point.
6. Select **Group 0**, click **SIP Method** and in the **Method** field, type INVITE.
7. Select the IM icon representing IM-SSF and in the **Properties** tab, type the address of the destination online charging application server in the **Alias** field and click **Apply**.

   The address of the application server must match the one set for the SCCP Local SSN.

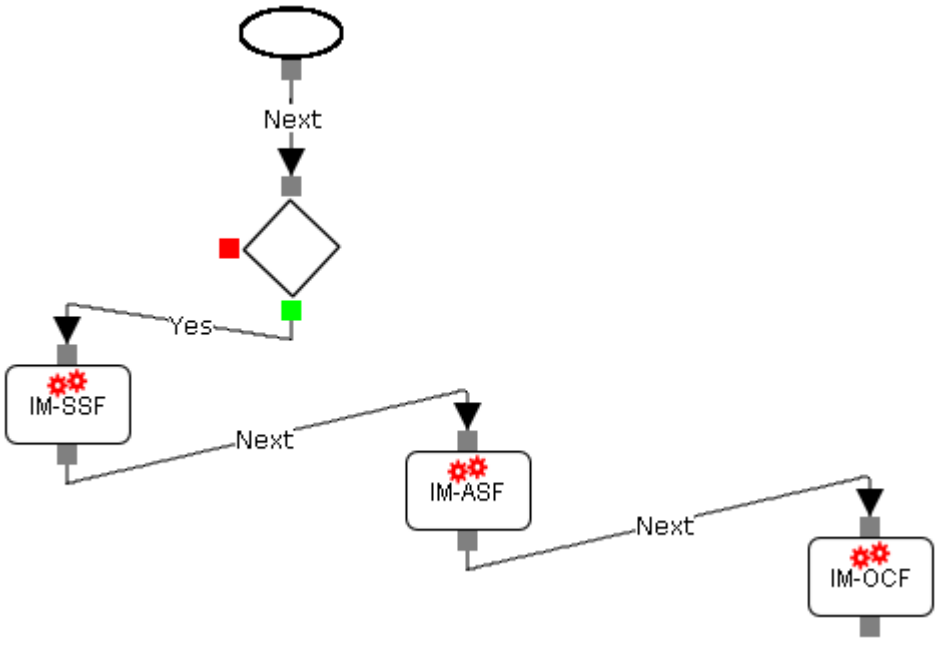
   For more information, see the discussion on configuring IM-SSF in *Oracle Communications Service Broker Modules Configuration Guide*.
8. Select the IM icon representing IM-ASF and in the **Properties** tab, type the address of the destination SIP application server in the **Alias** field and click **Apply**.

   Use the format: `sip:`*<ip address>*
9. Select the IM icon representing IM-OCF and in the **Properties** tab, type the address of the destination charging server in the **Alias** field and click **Apply**.

   Use the format: `sip:`*<ip address>*

   Figure A–3 shows the shows the orchestration logic flow for IMS service interaction.

Figure A–3 IMS Service Interaction Flow



A.2.2.2 Viewing the IMS Service Interaction Source Code

Clicking **Source** displays the XML code of the flow you created graphically.

Figure A–4 shows the source code of the IMS service interaction flow.

Figure A–4 IMS Service Interaction Source Code

```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>Invite</Method>
        <Extension>
          <ParentID>If Then Else1766</ParentID>
        </Extension>
      </SPT>
      <Extension>
        <YesNodeCNF>1</YesNodeCNF>
      </Extension>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IMSSF@ocsb</ServerName>
      <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
  <InitialFilterCriteria>
    <Priority>2</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <SIPHeader>
        <Header>x-wcs-history</Header>
        <Content>id=1.*</Content>
      </SIPHeader>
    </SPT>
  </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:IM-ASF@ocsb</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <ServiceInfo/>
    <Extension>
      <UnidirectionalGroup/>
    </Extension>
  </ApplicationServer>
</InitialFilterCriteria>
<InitialFilterCriteria>
  <Priority>3</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <SIPHeader>
        <Header>x-wcs-history</Header>
        <Content>id=2.*</Content>
      </SIPHeader>
    </SPT>
  </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:IMOCF@ocsb</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <ServiceInfo/>
    <Extension>
      <UnidirectionalGroup/>
    </Extension>
  </ApplicationServer>
</InitialFilterCriteria>
</IFCs>
```

A.2.3 Forcing Back to Back

By default, when the OE receives a **302 Moved Temporarily** response from an application, the OE releases the session. When you want the OE to continue the session after receiving a **302 Moved Temporarily** response, you need to enforce the application that returned the **302 Moved Temporarily** response to work as a Back-to-Back (B2B) application.

The following flow demonstrates the case when a VPN service with which the OE communicates through the IM-ASF a **302 Moved Temporarily** response. The IM-ASF is set to force the session to continue to the online charging application.

A.2.3.1 Enforcing the Back-to-Back Flow

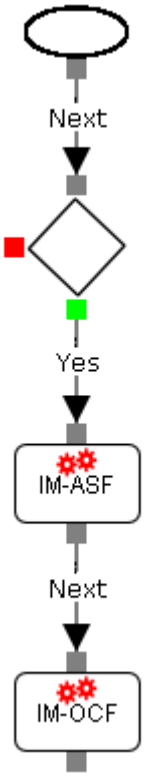
To enforce a B2B flow:

1. Drag the **Conditions** icon to canvas and draw a connecting line between **Start** and the icon.
2. Drag the **IM** icon representing the IM-ASF to the canvas and then drag the **IM** icon representing the IM-OCF to the canvas.
3. Draw a connector line from the green connection point on the **Conditions** icon to IM-ASF and and then draw a connector line from IM-ASF to IM-OCF.
4. Select the **Conditions** icon and in the **Actions** tab, click **New Group**.

   Group 0 is displayed under Trigger Point.
5. Select **Group 0**, click **SIP Method** and in the **Method** field, type INVITE.
6. Select the **IM** icon representing IM-ASF and in the **Actions** tab, do the following:
   - Select the **Back to Back** checkbox.
   - In the **Response** field, type **302**.

   The session is forced to continue to the IM-OCF. Figure A–5 shows a B2B flow.

Figure A–5 Forced Back to Back Flow



A.2.3.2 Viewing the FB2B Source Code

Clicking **Source** displays the XML code of the flow you created graphically.

Figure A–6 shows the source code of the B2B flow.

Figure A–6 Forced Back to Back Source Code

```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>Invite</Method>
        <Extension>
          <ParentID>If Then Else1231</ParentID>
        </Extension>
      </SPT>
      <Extension>
        <YesNodeCNF>1</YesNodeCNF>
      </Extension>
    </TriggerPoint>
    <ApplicationServer>
     <ServerName>sip:IM-ASF@ocsb</ServerName>
     <DefaultHandling>0</DefaultHandling>
     <ServiceInfo/>
     <Extension>
       <BackToBack>
         <response>302</response>
       </BackToBack>
       <UnidirectionalGroup/>
     </Extension>
   </ApplicationServer>
 </InitialFilterCriteria>
 <InitialFilterCriteria>
   <Priority>2</Priority>
   <TriggerPoint>
     <ConditionTypeCNF>0</ConditionTypeCNF>
     <SPT>
       <ConditionNegated>0</ConditionNegated>
       <Group>0</Group>
       <SIPHeader>
          <Header>x-wcs-history</Header>
          <Content>id=1.*</Content>
        </SIPHeader>
      </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IMOCF@ocsb</ServerName>
      <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

A.2.4 Choosing between Two Execution Paths

In the following use case, if the session is originating, the OE routes the session to the VPN application server and then to the online charging application server. If the session is terminating, the OE routes the session directly to the online charging application server.

A.2.4.1 Building a Choosing between Two Execution Paths Flow

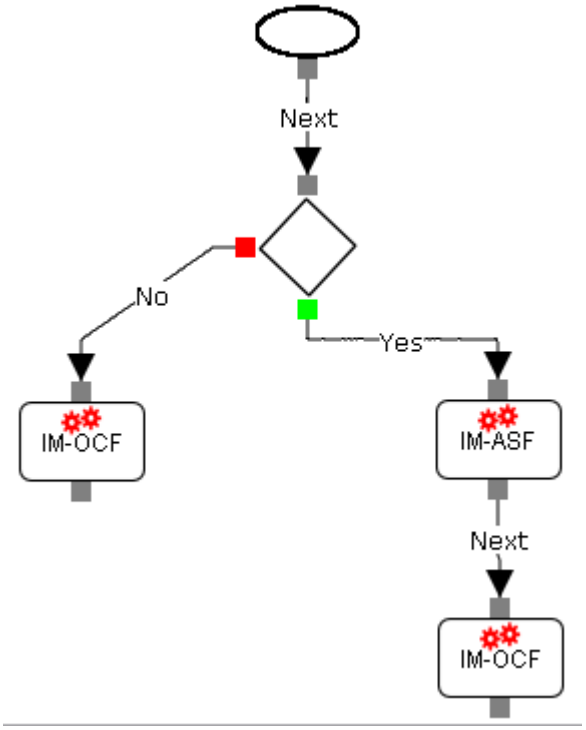
To build a flow that continues the session when conditions are not met:

1. Drag the **Conditions** icon to canvas and draw a connecting line between **Start** and the icon.
2. Drag the **IM** icon representing the IM-ASF to the canvas and draw a connecting line from the green connection point on the **Conditions** icon to the **IM** icon.
3. Drag the **IM** icon representing the IM-OCF to the canvas and draw a connecting line between IM-ASF and IM-OCF.
4. Drag another **IM** icon representing the IM-OCF to the canvas and draw a connecting line from the red connection point on the **Conditions** icon to the IM-OCF you added in step 3.
5. Select the **Conditions** icon and in the **Actions** tab, click **New Group**.

   Group 0 is displayed under Trigger Point.
6. Select **Group 0** and click **SIP Header**.
7. In the right pane, type the following:
   - **Header** field: `x-wcs-session-case`
   - **Content** field: `x-wcs-session-case:orig`

   Figure A–7 shows the conditional flow. If the message header meets the conditions, it goes to the IM-ASF and then to IM-OCF. If the message header does not meet the conditions, it goes directly to IM-OCF.

Figure A–7 Flow Choosing between Two Execution Paths



A.2.4.2 Viewing the Source Code of a Flow Choosing between Two Execution Paths

Clicking **Source** displays the XML code of the flow you created graphically.

Figure A–8 and Figure A–9 show the source code of the conditional flow.

Figure A–8 Flow Choosing between Two Execution Paths

```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <SIPHeader>
          <Header>x-wcs-session-case</Header>
          <Content>x-wcs-session-case:orig</Content>
        </SIPHeader>
        <Extension>
          <ParentID>If Then Else2199</ParentID>
        </Extension>
        </SPT>
        <Extension>
          <YesNodeCNF>1</YesNodeCNF>
        </Extension>
      </TriggerPoint>
      <ApplicationServer>
        <ServerName>sip:IM-ASF@ocsb</ServerName>
        <DefaultHandling>0</DefaultHandling>
        <ServiceInfo/>
        <Extension>
          <UnidirectionalGroup/>
        </Extension>
      </ApplicationServer>
    </InitialFilterCriteria>
    <InitialFilterCriteria>
    <Priority>2</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <SIPHeader>
          <Header>x-wcs-history</Header>
          <Content>id=1.*</Content>
        </SIPHeader>
      </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IMOCF@ocsb</ServerName>
      <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
  <InitialFilterCriteria>
```

Figure A–9 Flow Choosing between Two Execution Paths (continued)

```
<Priority>3</Priority>
<TriggerPoint>
  <ConditionTypeCNF>1</ConditionTypeCNF>
  <SPT>
    <ConditionNegated>1</ConditionNegated>
    <Group>0</Group>
    <SIPHeader>
      <Header>x-wcs-session-case</Header>
      <Content>x-wcs-session-case:orig</Content>
    </SIPHeader>
  </SPT>
  <Extension>
    <YesNodeCNF>0</YesNodeCNF>
  </Extension>
</TriggerPoint>
<ApplicationServer>
  <ServerName>sip:IMOCF@ocsb</ServerName>
  <DefaultHandling>0</DefaultHandling>
      <ServiceInfo/>
      <Extension>
        <UnidirectionalGroup/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

A.3 Building an Orchestration Flow in Online Mediation Controller

When you create an orchestration logic using the Orchestration Studio, Service Broker stores the orchestration logic in SM-LSS as a part of the subscriber's profile. With Online Mediation Controller, subscriber profile data, including subscriber-specific orchestration logic, is stored in the Subscriber Store instead of SM-LSS.

To access the Subscriber Store, Service Broker uses the Subscriber Provisioning API. The API provides operations for adding and managing subscriber profile data in the store. The operation that adds a new subscriber to the Subscriber Store is called storeSubscriber. In the request body, you provide the orchestration logic in the iFC format.

To facilitate the process of creating of iFC, you can build the orchestration logic using the Orchestration Studio visual tools. Because the Orchestration Studio generates the iFC code of the orchestration logic, you can copy the iFC representation and paste it into the storeSubscriber request body.

To create an orchestration logic for a subscriber in Online Mediation Controller:

1. Create a subscriber profile in SM-LSS. For more information, see the discussion on configuring SM-LSS in *Oracle Communications Service Broker Modules Configuration Guide*.
2. Build an orchestration flow for the subscriber using the Orchestration Studio tools. See "Building an Orchestration Logic Flow" for more information.
3. To view the iFC code that the Orchestration Studio generated for the orchestration flow, click Source.
4. Copy the entire iFC code and paste it into the <ifc> element of the storeSubscriber request body.

For more information, see the discussion on the subscriber store API reference in *Oracle Communications Service Broker Subscriber Store User's Guide*.

B

Initial Filter Criteria

This appendix describes how you can set up the Initial Filter Criteria (iFC) using standard iFC elements and proprietary extensions supported by the Oracle Communications Service Broker Orchestration Engine (OE).

B.1 About the Initial Filter Criteria

The iFC is an XML-based IP Multimedia Subsystem (IMS) standard that you use to define the order in which the OE routes a session across applications. The following documents describe the standard iFC elements:

- ETSI TS 129 228 V7.11.0, IP Multimedia (IM) Subsystem Cx and Dx Interfaces
- 3GPP TS 29.328 V7.11.0, IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents, Release 7.

B.2 Setting Up the Initial Filter Criteria

The iFC defines the order in which the OE routes a session across applications. The routing is conditional. This means the OE routes the session to a specific application only when the session meets the criteria specified for that application. For example, you can specify that the OE routes the session to a Virtual Private Network application only when the session's Called Party Number begins with the asterisk (*).

A set of conditions that a session must meet and the application to which the OE routes the session is known as initial filter criteria. You enclose initial filter criteria in the `<InitialFilterCriteria>` element. You can create as many `<InitialFilterCriteria>` elements as you need.

In each `<InitialFilterCriteria>` element, you specify the following elements:

- `<TriggerPoint>`, which contains one or more conditions that must be met in order to route the session to a specific application. See "Setting Up a Trigger Point" for more information.
- `<Application>`, which defines the application to which the OE routes the session if all conditions are met. This element includes the definition of the application name and instructions for handling the session when the OE receives error responses from applications. See "Specifying an Application" for more information.
- `<Priority>`, which defines the priority of the iFC when conditions of multiple `<InitialFilterCriteria>` elements are met. See "Specifying a Priority" for more information.

B.2.1 Setting Up a Trigger Point

A trigger point consists of one or more conditions that the session must meet in order to be routed to a specific application. In the iFC, each condition is called Service Point Trigger (SPT). To set up an SPT, you use the <SPT> element.

You can specify conditional statements using AND and OR conditions between SPTs. For example, you might specify that the OE routes the session to an application if the following condition is met: (SPT1 OR SPT2) AND (SPT3 OR SPT4).

To set up conditional statements, you need to group SPTs. Then you specify the relationship between groups of SPTs and members of each group.

For example, in the statement (SPT1 OR SPT2) AND (SPT3 OR SPT4), SPT1 and SPT2 belong to one group. SPT3 and SPT4 belong to another group. The relationship between group members is OR. The relationship between the groups is AND.

B.2.1.1 Grouping SPTs and Specifying Relationship Between Groups and Group Members

To group SPTs, you specify an integer which represents the group to which the SPT belongs, in the <Group> element. You place this element under the <SPT> element. The SPTs whose <Group> element is set to the same number belong to the same group.

To specify the relationship between groups of SPTs and members of each group, you use the <ConditionTypeCNF> element placed under the <TriggerPoint> element. You can set the <ConditionTypeCNF> element to one of the following values:

- 0: the relationship between group members is AND while the relationship between groups is OR. For example, (SPT1 AND SPT2) OR (SPT3 AND SPT4).
- 1: the relationship between group members is OR while the relationship between groups is AND. For example: (SPT1 OR SPT2) AND (SPT3 OR SPT4).

B.2.1.2 Specifying Conditions

You can use the following criteria as conditions that the session must meet:

- <Method>, which defines a SIP method used to initiate a call. For example, you can specify that the OE triggers an application only if the method is INVITE.
- <SIPHeader>, which consists of the following elements:
  - <Header>, which defines the name of the header that you want to check. You can specify any standard SIP header as well as any custom header.
  - <Content>, which defines the contents of the header
- <RequestURI>, which contains the URI of the destination application server defined in the session request.
- <Line> and <Content>, which contain the session type to be communicated between the two end parties. The OE uses these criteria when the content type of the message body is "application/sdp". The OE checks whether the text in each line of the incoming message body matches the text that you specified in <Line> tag. After the OE found the line, the OE checks whether this line contains the text specified in the <Content> tag.

You can negate the condition using the <ConditionNegated> element. Use a negated condition to sepcify that a condition is met for values other than the ones specified in the condition. For example, you can route a session to a module only if the SIP Method is other than INVITE. To negate a condition, set <ConditionNegated> to 1. Otherwise, set <ConditionNegated> to 0.

The following code shows a scenario in which the iFC defines SPTs as follows:

Table B–1 SPTs Definitions in the Sample iFC

SPT	Condition to Be Checked	Group
SPT 1	Method of the session is INVITE.	1
SPT 2	SessionCase of the session is 0.	1
SPT 3	Method of the session is INVITE.	2
SPT 4	From header of the session does not contain "joe".	2
SPT 5	RequestURI header of the session contains "sip:destination_server1@oracle.com".	3
SPT 6	SessionCase of the session is 2.	3

The OE routes the session to the application if the following condition is met: (INVITE OR SESSIONCASE=0) AND (INVITE OR FROM != "Joe") AND (requestURI="sip:destination_server1@oracle.com" OR SessionCase=2)

```
<iFCs>
  <InitialFilterCriteria>
    <Priority>0</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>

      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>1</Group>
        <Method>INVITE</Method>
      </SPT>

      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>1</Group>
        <SessionCase>0</SessionCase>
      </SPT>

      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>2</Group>
        <Method>INVITE</Method>
      </SPT>

      <SPT>
        <ConditionNegated>1</ConditionNegated>
        <Group>2</Group>
        <SIPHeader>
          <Header>From</Header>
          <Content>"joe"</Content>
        </SIPHeader>
      </SPT>

      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>3</Group>
        <RequestURI>sip:destination_server1@oracle.com</RequestURI>
      </SPT>
```

```
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>3</Group>
        <SessionCase>0</SessionCase>
      </SPT>

    </TriggerPoint>

  <ApplicationServer>
    <ServerName>sip:app1@oracle.com</ServerName>
    <DefaultHandling>0</DefaultHandling>
  </ApplicationServer>
</InitialFilterCriteria>
```

B.2.2 Specifying an Application

When the conditions set in the <TriggerPoint> element are met, the OE routes the session to the application that you specify in the <ApplicationServer> element.

In this element, you define the following mandatory elements:

- <ServerName>, which defines the SIP URL of an IM to which the OE routes the session
- <Default Handling>, which defines whether or not the OE releases a session if an application cannot be reached. You can set <Default Handling> to one of the following values:
  - 0: to continue the session
  - 1: to terminate the session

The following code shows a scenario in which the OE routes the session to the IM whose SIP URI is sip:as2@192.168.1.140:5060. In this example, the OE continues the session if the application cannot be reached.

```
<ApplicationServer>
  <ServerName>sip:app1@oracle.com</ServerName>
  <DefaultHandling>0</DefaultHandling>
</ApplicationServer>
```

See "Continuing or Releasing a Session" for more information about configuring of the default handling.

B.2.3 Specifying a Priority

In some cases, conditions defined in several different filter criteria can be met. To enable the OE to choose a specific filter criteria, you can define a filter criteria's priority.

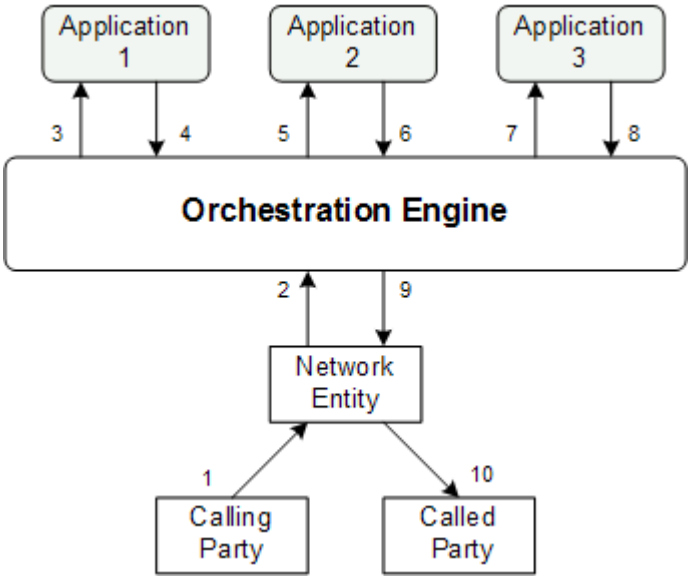
The higher the rule's priority number, the lower priority the filter criterion has. This means that a filter criterion with a higher value of priority number is assessed after the filter criteria with a smaller priority number has been assessed. 0 (zero) means the highest priority. 100 means the lowest priority.

The same priority cannot be assigned to more than one initial filter criterion.

B.3 Specifying the Order of Message Routing

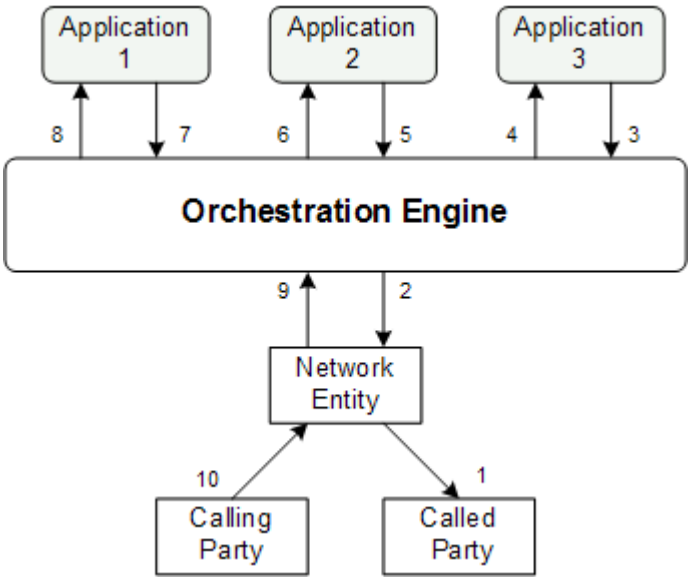
The iFC defines how the OE routes messages generated by the calling party. For example, you can set an initial INVITE to be routed from Application 1 to Application 2 to Application 3. Figure B–1 shows the order in which the OE routes an INVITE message from a calling party to a called party.

Figure B–1 Routing an INVITE Message from a Calling Party



The iFC does not specify how the OE routes messages received from a called party. By default, when a called party generates a message (for example, an OK response to an INVITE message), the OE routes this message in the reverse order, from Application 3 to Application 2 to Application 1. Figure B–2 shows the order in which the OE routes an OK response from a called party to a calling party.

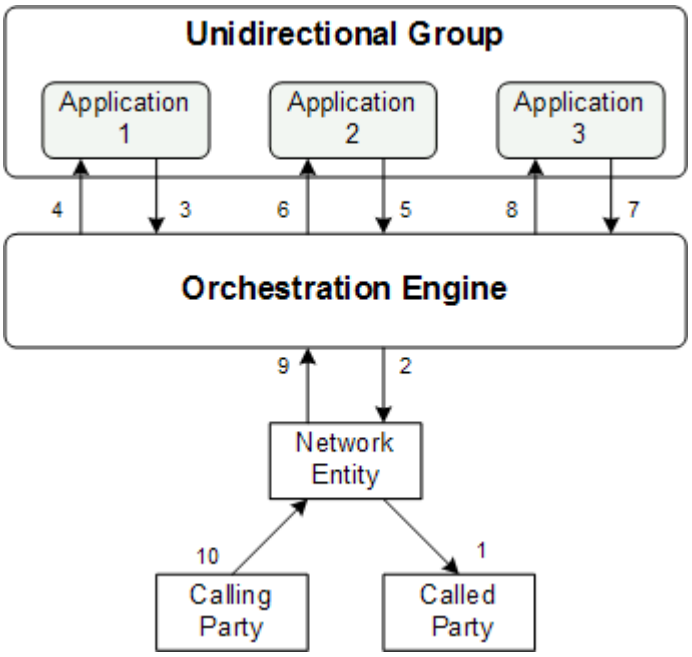
Figure B–2 Routing an OK Message from a Called Party



When an application in the orchestration chain depends on the information generated by a previous application, you might need to route all messages, including those generated by a calling party and those generated by a called party, in the same order. For example, you might need a message to be first routed to an online charging application and then to a bill shock application. In this case, a bill shock application can perform certain actions based on the information generated by the online charging application.

To allow the OE to route all messages across applications in the same order, you need to group these applications in a unidirectional group. Figure B–3 shows how the OE routes a message generated by a called party through applications in a unidirectional group.

Figure B–3 Routing an OK Message from a Called Party in a Unidirectional Group



When grouping applications into unidirectional groups, you must observe the following limitations:

- You can group only those applications that run consecutively. For example, on Figure B–3, you can group Application 1 and Application 2. However, you cannot group Application 1 and Application 3 because they do not run consecutively.
- Each application in a unidirectional group must be configured as a Back-to-Back (B2B) application (see "Continuing or Releasing a Session" for more information about B2B applications).

You use the `<UnidirectionalGroup>` element to assign an application to a unidirectional group. Because this element is an extension to the standard iFC, you need to put it under the `<Extension>` element.

You set `<UnidirectionalGroup>` to the identifier of a group. This identifier must be an integer. You can have as many unidirectional groups as you need.

The following code shows an example of how you can assign two applications to the same unidirectional group (the `<UnidirectionalGroup>` is bolded).

```
<iFCs>
  <InitialFilterCriteria>
```

```
    <Priority>2</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
      </SPT>
    </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:as2@192.168.1.140:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <ServiceInfo></ServiceInfo>
    <Extension>
      <UnidirectionalGroup>1</UnidirectionalGroup>
      <ForceB2B/>
    </Extension>
  </ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
  <Priority>3</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <Method>INVITE</Method>
    </SPT>
  </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:as3@192.168.1.141:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <ServiceInfo></ServiceInfo>
      <Extension>
        <UnidirectionalGroup>1</UnidirectionalGroup>
        <ForceB2B/>
      </Extension>
    </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

Note: Service Broker does not support unidirectional groups when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

B.4 Providing Additional Information to an Application

You can configure the OE to send additional information to an application by using the `<ServiceInfo>` element. You need to place this element under the `<ApplicationServer>`.

The OE adds the text specified in `<ServiceInfo>` to the body of the message that the OE forwards to the application.

The following code shows an example of how you can specify additional information to be sent to the application (the <ServiceInfo> element is bolded).

```
<ApplicationServer>
  <ServerName>sip:as@192.168.0.1:5060</ServerName>
  <DefaultHandling>0</DefaultHandling>
  <ServiceInfo>application-specific information</ServiceInfo>
</ApplicationServer>
```

B.5 Continuing or Releasing a Session

When the OE receives a final response from an application (that is a 3xx, 4xx, or 5xx response), the OE can either release or continue the session. You specify the action that the OE performs using the following elements:

- <DefaultHandling>

  See "Specifying an Application" for more information on this element.

- <ForceB2B>

  This element is an extension to the standard iFC and must be placed under the <Extension> element.

  Using the <ForceB2B> element, you can specify conditions that force the OE to continue the session when the OE receives a response (such as 302 Moved Temporarily or 400 Bad Request) from an application. These conditions are based on response codes received by the OE from an application. You can force session continuity on specific response codes using the <response> element. You need to add this element under the <ForceB2B> element.

  The following code shows a scenario in which the OE continues the session only when the application returns either the response code 302 Moved Temporarily or 400 Bad Request.

  ```
  <Extension>
    <ForceB2B>
      <response>302</response>
      <response>400</response>
    <ForceB2B>
  </Extension>
  ```

  If you add an empty <ForceB2B> element (that is you do not explicitly specify any response codes), the OE continues the session only when the OE receives a 302 Moved Temporarily code. The following code shows such a scenario.

  ```
  <Extension>
    <ForceB2B/>
  </Extension>
  ```

For a more fine-grained configuration, you can use <ForceB2B> in conjunction with <DefaultHandling>. The following example shows a scenario in which the OE releases a session if an application sends an error message (<DefaultHandling> is set to 1). However, if the error is 408 Request Timed Out, the OE continues the session (the <response> element is bolded):

```
<DefaultHandling>1</DefaultHandling>
<Extension>
  <ForceB2B>
    <response>408</response>
  <ForceB2B>
</Extension>
```

By default, the OE continues the session if the 302 Moved Temporarily response is received. Otherwise, the OE releases the session.

Note: Service Broker does not support the default handling and force B2B features when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

B.6 Triggering Applications Based on the Status of the Previous Application

The status of an application is determined by a request or response that the application returns to the OE. You can set up any application in the orchestration chain to run only if the previous application in the orchestration chain has a specific status that the application returns a specific request or response.

For example, you can specify that the OE triggers an application only if the previous application returns INVITE to the OE. Alternatively, you can define that the OE triggers an application only if the previous application returns the response code 302 Moved Temporarily.

The information about a response or request which the previous application returns is stored in the **x-wcs-history** header of the message. This header has the following format:

```
x-wcs-history: id=application_identifier; status=status_code
```

The parameters are defined as follows:

- **id**

  This is an identifier of the previous application in the orchestration chain. The application sets this parameter to the value of the `<Priority>` element of the iFC that triggered that application.

- **status**

  This is the status of the previous application in the orchestration chain.

  The application status is an integer. It represents the message that the application returns to the OE. This message can contain one of the following:

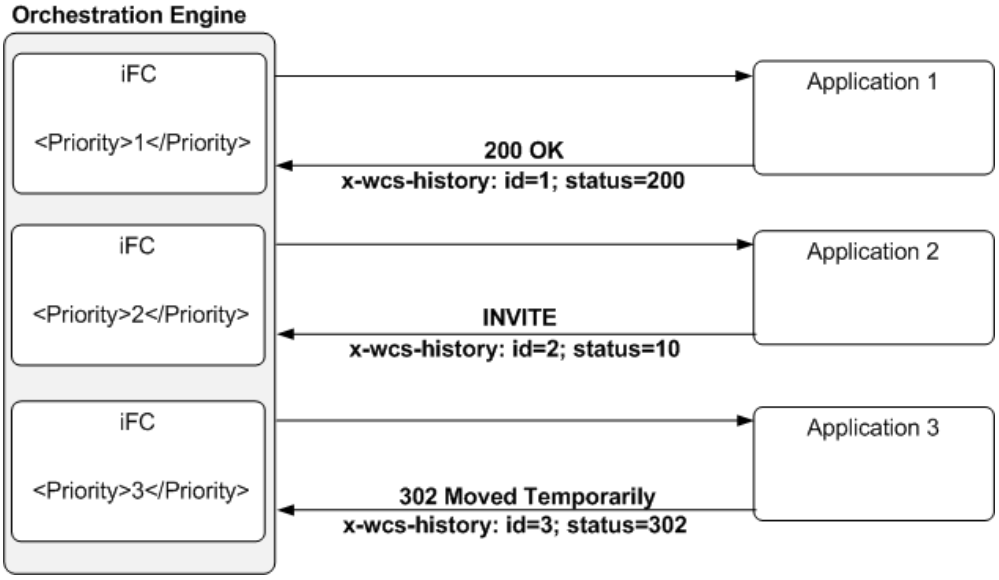
  - Response code, such as 200 or 302. In this case, the application sets the **status** parameter to the response number.
  - Request name, such as INVITE or SUBSCRIBE. In this case, the OE maps the name of a request to an integer as you configured using the OeHistoryMBean (see "Mapping Request Names to Status" for more information).

Figure B–4 shows a scenario in which the OE routes the session as follows:

1. The OE routes the session to Application 1. The `<Priority>` element of the iFC that triggers this application is 1. Application 1 returns to the OE the response code 200 OK. Application 1 sets the **x-wcs-history** header as follows:
   - id=1
   - status=200

2. Then the OE routes the session to Application 2. The `<Priority>` element of the iFC that triggers this application is 2. Application 2 returns to the OE the INVITE request. This request is mapped to 10 using OeHistoryMBean. Application 2 updates the **x-wcs-history** header as follows:
   - id=2
   - status=10
3. Finally, the OE routes the session to Application 3. The `<Priority>` element of the iFC that triggers this application is 3. Application 3 returns to the OE the response code 302 Moved Temporarily. Application 3 updates the **x-wcs-history** header as follows:
   - id=3
   - status=302

Figure B–4 Returning Information about Response from the Previous Application



B.6.1 Mapping Request Names to Status

The **status** parameter of the **x-wcs-history** header is the integer that describes the message that an application returns to the OE. If an application returns a request (for example, an INVITE), you need to map this response to an integer using the **OeHistoryMBean**. For example, you can map an INVITE message to 10.

To map a response:

1. Create an instance of **RequestStatusCodesMBean** by invoking the following operation of **OeHistoryMBean**:

   ```
   ObjectName createRequestStatusCodes()
   ```

2. Create an instance of **RequestStatusCodeMBean** by invoking the following operation of **RequestStatusCodesMBean**:

   ```
   ObjectName createRequestStatusCode()
   ```

3. Set the attributes of **RequestStatusCodeMBean** as follows:

- Set the **Request** attribute of **RequestStatusCodeMBean** to the message that you want to map.
- Set the **StatusCode** attribute of **RequestStatusCodeMBean** to the integer to which you want to map the message that the session contains.

See "Java MBeans Reference" for more information about these MBeans.

B.6.2 Triggering an Application

You can set up an application to run depending on the contents of the **x-wcs-history** header.

To evaluate the **x-wcs-history** header, you use the `<Header>` and `<Content>` elements. For example, the following code shows a scenario in which application sip:as1@192.168.1.140:5060 is triggered by the iFC whose `<Priority>` element is set to 1. The application returns an INVITE to the OE. (The example assumes that INVITE messages are mapped to 10 using **OeHistoryMBean**.)

Application sip:as2@192.168.1.141:5060 runs only if the **x-wcs-history** header of the message returned from the previous application is set as follows:

- id=1
- status=10

```
<iFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
      </SPT>
    </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:as1@192.168.1.140:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
  </ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
  <Priority>2</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <SIPHeader>
        <Header>x-wcs-history</Header>
        <Content>id=1;status=10</Content>
      </SIPHeader>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
    </SPT>
  </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:as2@192.168.1.141:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
    </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

B.7 Merging Conditional Routes

If you build an orchestration logic in which the OE routes the session to different applications based on certain conditions, you can merge different conditional routes after they passed the respective applications.

For example, you can build an orchestration logic that routes the session to an IM-ASF if the condition is met or to an IM-OCF if the condition is not met. After the session passed the IM-ASF or IM-ASF, the OE routes the session to IM-WS.

You specify the applications from which you want to merge conditional routes by defining the value of the `<Priority>` element of the iFC that triggered the application. The OE stores the value of `<Priority>` in the **id** parameter of the **x-wcs-history** header. Therefore, for each application from which you want to merge the route, you need to create an SPT that checks whether the **x-wcs-history** contains a specific **id**.

The following example contains the definitions for the following applications:

- IM-ASF, which receives the session if it contains a **SIP INVITE** message (see the `<ConditionNegated>` element set to 0). This application has the `<Priority>` set to 1.
- IM-OCF, which receives the session if it does not contain a **SIP INVITE** message (see the `<ConditionNegated>` element set to 1). This application has the `<Priority>` set to 2.
- IM-WS, which receives the session after it passed IM-ASF or IM-OCF. This application checks whether the **id** parameter of the **x-wcs-history** header is set to 1 or 2 that is whether the session passed through either IM-ASF or IM-OCF. **.*** in the `<Content>` element means that IM-ASF and IM-OCF might have any status.

```
<InitialFilterCriteria>
   <Priority>1</Priority>
   <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
         <ConditionNegated>0</ConditionNegated>
         <Group>0</Group>
         <Method>INVITE</Method>
      </SPT>
   </TriggerPoint>
   <ApplicationServer>
      <ServerName>sip:IM-ASF.IMASF@ocsb.com</ServerName>
      <DefaultHandling>0</DefaultHandling>
   </ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
   <Priority>2</Priority>
   <TriggerPoint>
      <ConditionTypeCNF>1</ConditionTypeCNF>
      <SPT>
         <ConditionNegated>1</ConditionNegated>
         <Group>0</Group>
         <Method>INVITE</Method>
      </SPT>
   </TriggerPoint>
   <ApplicationServer>
      <ServerName>sip:IM-OCF.IMOCF@ocsb.com</ServerName>
      <DefaultHandling>0</DefaultHandling>
   </ApplicationServer>
</InitialFilterCriteria>
```

```
<InitialFilterCriteria>
  <Priority>3</Priority>
   <TriggerPoint>
      <ConditionTypeCNF>1</ConditionTypeCNF>
      <SPT>
         <ConditionNegated>0</ConditionNegated>
         <Group>0</Group>
         <SIPHeader>
            <Header>x-wcs-history</Header>
            <Content>id=1;.*</Content>
         </SIPHeader>
      </SPT>
      <SPT>
         <ConditionNegated>0</ConditionNegated>
         <Group>0</Group>
         <SIPHeader>
            <Header>x-wcs-history</Header>
            <Content>id=2;.*</Content>
         </SIPHeader>
      </SPT>
   </TriggerPoint>
   <ApplicationServer>
      <ServerName>sip:IMWS.IMWS@ocsb.com</ServerName>
      <DefaultHandling>0</DefaultHandling>
   </ApplicationServer>
</InitialFilterCriteria>
```

B.8 Triggering Applications Based on the Previous Session Route

You can set up an application to run based on the previous route of the session. For example, you can set up a bill shock prevention application to run only if the session was previously routed to an online charging application based on the **From** header.

To enable applications in the orchestration chain to get the information about applications through which the session is routed, you need to tag the session when it passes through an application. For example, if the OE routes the session to an online billing application, you can tag the session with the tag OnlineBillingTriggered. This allows next applications in the orchestration chain to be aware of the online billing application was triggered.

Then you can set up one of the next applications in the chain (such as a bill shock prevention application) to run only if the session contains the OnlineBillingTriggered tag.

As the session passes through applications in the orchestration chain, all tags are accumulated. Therefore, any subsequent application in the orchestration chain can be aware of those tagged applications which were previously triggered.

The following sections explain how to tag a session and trigger an application only if the session contains the specified tags.

Note: Service Broker does not support this feature when the Diameter-based orchestration mode is enabled. For more information on the Diameter-based orchestration mode, see the discussion on improving performance in Diameter-only environments in *Oracle Communications Service Broker Online Mediation Controller Implementation Guide*.

B.8.1 Tagging a Session

To tag a session, you use the <Tags> element. This element is an extension to the standard iFC. You need to place the <Tags> element under the <Extension> element.

You can add into <Tags> as many tags as you need. The tags must be separated by comma.

The following code shows a scenario in which the OE routes the session to an online billing application and tags the session with the OnlineBillingTriggered tag (the <Tags> element is bolded).

```
<iFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <SIPHeader>
          <Header>From</Header>
          <Content>"joe"</Content>
        </SIPHeader>
      </SPT>
    </TriggerPoint>
  <ApplicationServer>
    <ServerName>sip:as2@192.168.1.140:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <Extension>
      <Tags>OnlineBillingTriggered</Tags>
    </Extension>
  </ApplicationServer>
</InitialFilterCriteria>
```

The OE adds the contents of the <Tags> element to the **x-wcs-tags** header of the message that the OE routes to the next application in the orchestration chain. This header has the following format:

```
x-wcs-tags: comma_separated_tags_accumulated_from_all_previous_applications
```

The application returns the message to the OE with the **x-wcs-tags** header intact. As the message passes through applications in the orchestration chain, the **x-wcs-tags** header accumulates the contents of all <Tags> elements defined for applications in the chain.

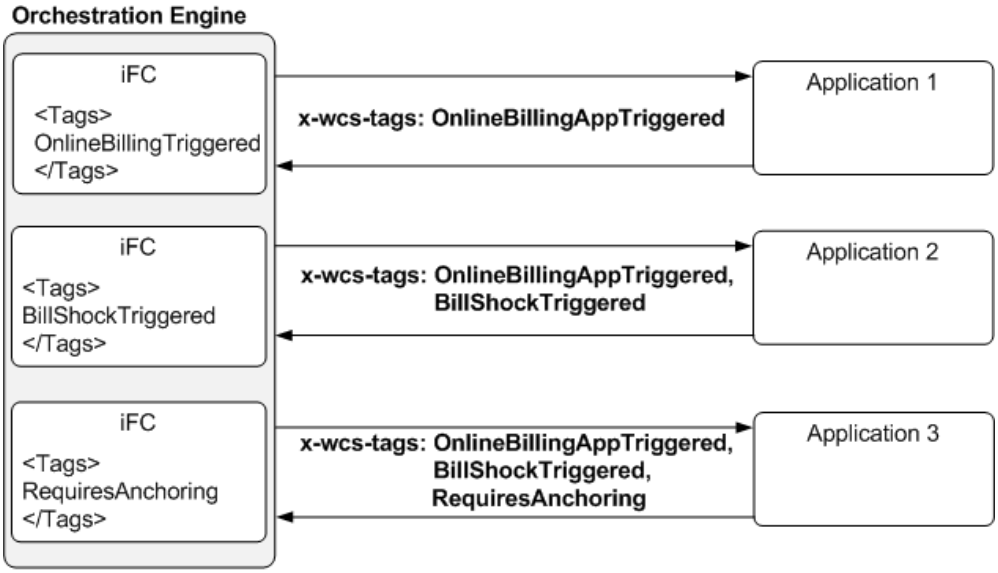
Figure B–5 shows a scenario in which the OE routes a session from Application 1 to Application 2 to Application 3. The session is tagged as follows:

- When the OE routes the to Application 1, the tag OnlineBillingTriggered is added.
- When the OE routes the session to Application 2, the tag BillShockTriggered is added.

- When the OE routes the session to Application 3, the tag RequiresAnchoring is added.

Because the **x-wcs-tags** header accumulates added tags, after triggering Application 3, the header contains "OnlineBillingTriggered, BillShockTriggered, RequiresAnchoring".

Figure B–5 Accumulating Session Tags



B.8.2 Triggering an Application

You can set up an application to run depending on whether the **x-wcs-tags** header of the session contains specific tags.

To evaluate the **x-wcs-tags** header, you use the `<Header>` and `<Content>` elements. For example, the following code shows a scenario in which the session is tagged as follows:

- When the OE routes the session to sip:as1@192.168.1.140:5060, the tag OnlineBillingTriggered is added.
- When the OE routes the session to sip:as2@192.168.1.141:5060, the tag BillShockTriggered is added.
- When the OE routes the session to sip:as2@192.168.1.141:5060, no tag is added.

The OE triggers sip:as3@192.168.1.143:5060 only if the **x-wcs-tags** header of the session contains both OnlineBillingTriggered and BillShockTriggered.

```
<iFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
      </SPT>
    </TriggerPoint>
```

```
  <ApplicationServer>
    <ServerName>sip:as1@192.168.1.140:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
    <Extension>
      <Tags>OnlineBillingTriggered</Tags>
    </Extension>
  </ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
  <Priority>2</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <SIPHeader>
        <Header>From</Header>
        <Content>"joe"</Content>
    </SPT>
  </TriggerPoint>
<ApplicationServer>
  <ServerName>sip:as2@192.168.1.141:5060</ServerName>
  <DefaultHandling>0</DefaultHandling>
    <Extension>
      <Tags>BillShockTriggered</Tags>
    </Extension>
</ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
  <Priority>3</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <Method>INVITE</Method>
    </SPT>
  </TriggerPoint>
<ApplicationServer>
  <ServerName>sip:as2@192.168.1.141:5060</ServerName>
  <DefaultHandling>0</DefaultHandling>
</ApplicationServer>
</InitialFilterCriteria>

<InitialFilterCriteria>
  <Priority>4</Priority>
  <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <SIPHeader>
        <Header>x-wcs-tags</Header>
        <Content>OnlineBillingTriggered, BillShockTriggered</Content>
      </SIPHeader>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
    </SPT>
  </TriggerPoint>
  <ApplicationServer>
```

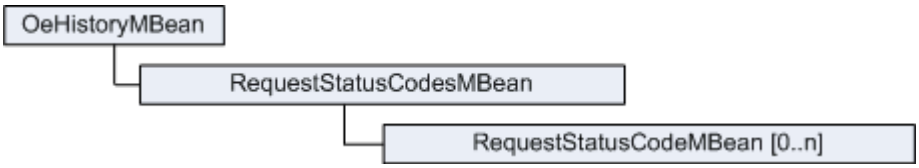
```
    <ServerName>sip:as3@192.168.1.143:5060</ServerName>
    <DefaultHandling>0</DefaultHandling>
  </ApplicationServer>
  </InitialFilterCriteria>
</IFCs>
```

Java MBeans Reference

You can use **OeHistoryMBean** and its child MBeans to map a response message to an integer. For example, you can map an INVITE message to 10.

Figure B–6 shows the hierarchy of the **OeHistoryMBean**.

Figure B–6 OeHistoryMBean Hierarchy



OeHistoryMBean

OeHistoryMBean enables you to map a response message to an integer. For example, you can map an INVITE message to 10. This value is stored in the **x-wcs-history** header and enables you to check the status of a previous application in the orchestration chain. See "Triggering Applications Based on the Status of the Previous Application" for more information.

Object Name

com.convergin:Type=OEHistory,Version=*MBean_Version*,Location=AdminServer,Name=oe_instance.OE_oe_instance_*MBean_Version*

Factory Method

Created automatically.

Attributes

int DefaultRequestStatusCode
Specifies the default value of the **status_code** parameter that the OE receives in the **x-wcs-histor**y header.

Operations

ObjectName createRequestStatusCodes()
Creates an instance of RequestStatusCodesMBean.

void destroyRequestStatusCodes()
Destroys an instance of RequestStatusCodesMBean.

ObjectName[] lookupRequestStatusCodes()
Gets an array of references to the instances of RequestStatusCodesMBean.

RequestStatusCodesMBean

RequestStatusCodesMBean is the root MBean for instances of RequestStatusCodeMBean. Each instance of RequestStatusCodeMBean enables you to map a single response to an integer.

Object Name

com.convergin:Type=RequestStatusCodes,Version=*MBean_Version*,Location=AdminServer,Name=oe_instance.OE_oe_instance_*MBean_Version*

Factory Method

OeHistory.createRequestStatusCodes()

Attributes

None

Operations

ObjectName createRequestStatusCode()
Creates an instance of RequestStatusCodeMBean.

void destroyRequestStatusCode()
Destroys an instance of RequestStatusCodeMBean.

ObjectName[] lookupRequestStatusCode()
Gets an array of references to the instances of RequestStatusCodeMBean.

RequestStatusCodeMBean

RequestStatusCodeMBean enables you to map a single response to an integer. You need to create a separate instance of RequestStatusCodeMBean for each response.

Object Name

com.convergin:Type=RequestStatusCode,Version=*MBean_Version*,Location=AdminServer,Name=oe_instance.String

Factory Method

RequestStatusCodes.createRequestStatusCode()

Attributes

string Request
Specifies the message that the session contains.

int StatusCode
Specifies the integer to which you want to map the message that the session contains.

Operations

None