Oracle® Communications Service Controller

Orchestration User's Guide

Release 6.2

F18711-02

April 2020

Oracle Communications Service Controller Orchestration User's Guide, Release 6.2

F18711-02

Contents

Preface v

Audience v
Documentation Accessibility v

1 Application Orchestration Overview

About Application Orchestration 1-1
About Orchestration Logic 1-2
About Subscriber Profile Receivers and Orchestration Logic Processors 1-3

2 Configuring the Orchestration Engine

Setting Up the Orchestration Engine 2-1
Configuring General Parameters 2-1
Configuring Static Route OLP Parameters 2-3
Configuring HSS OLP Parameters 2-3
Configuring Monitoring Parameters 2-4
Routing a Session through Non-Configured Applications 2-4

3 Invoking Applications Based on the Previous Session Route

About Invoking Applications Based on the Previous Session Route 3-1
Tagging a Session 3-1
Checking Tags in a Session 3-2

4 Defining the Orchestration Order of Messages Sent by a Called Party

About the Orchestration Order 4-1

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

About the Orchestration Engine Behavior on Receiving Responses from the Application 5-1
Defining the Orchestration Engine Behavior on Receiving an Error from the Application 5-1
Defining the Orchestration Engine Behavior on Receiving a Response from the Application 5-2

A Use Cases

About the Use Cases A-1
Service Orchestration A-1
IN Service Interaction A-1
IMS Service Interaction A-2
Forcing Back to Back A-4
Choosing between Two Execution Paths A-5

B Initial Filter Criteria

About the Initial Filter Criteria B-1
Setting Up the Initial Filter Criteria B-1
Setting Up a Trigger Point B-2
Grouping SPTs and Specifying Relationship Between Groups and Group Members B-2
Specifying Conditions B-2
Specifying an Application B-4
Specifying a Priority B-4
Specifying the Order of Message Routing B-4
Providing Additional Information to an Application B-7
Continuing or Releasing a Session B-8
Triggering Applications Based on the Status of the Previous Application B-9
Mapping Request Names to Status B-10
Triggering an Application B-11
Merging Conditional Routes B-12
Triggering Applications Based on the Previous Session Route B-13
Tagging a Session B-14
Triggering an Application B-15
Java MBeans Reference B-17
OeHistoryMBean B-18
RequestStatusCodesMBean B-19
RequestStatusCodeMBean B-20

Preface

This document provides a description of Oracle Communications Service Controller 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

Documentation Accessibility

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

Access to Oracle Support

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

1 Application Orchestration Overview

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

About Application Orchestration

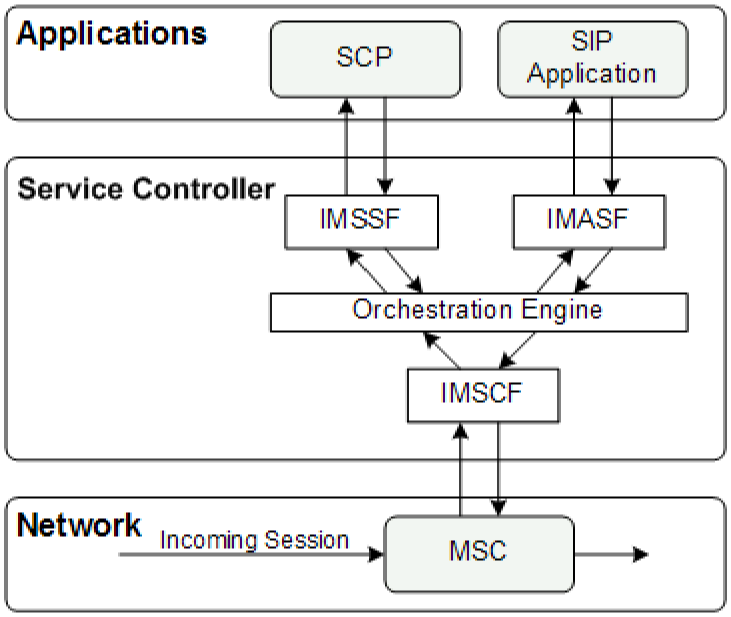
Orchestration is the ability of Service Controller to route a session through various applications. Service Controller routes a session sequentially, from one application to another. Each application executes a certain business logic. Every application applies a service on the session before Service Controller 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 Controller, 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.
- Pre-defined list of applications that Service Controller should invoke.

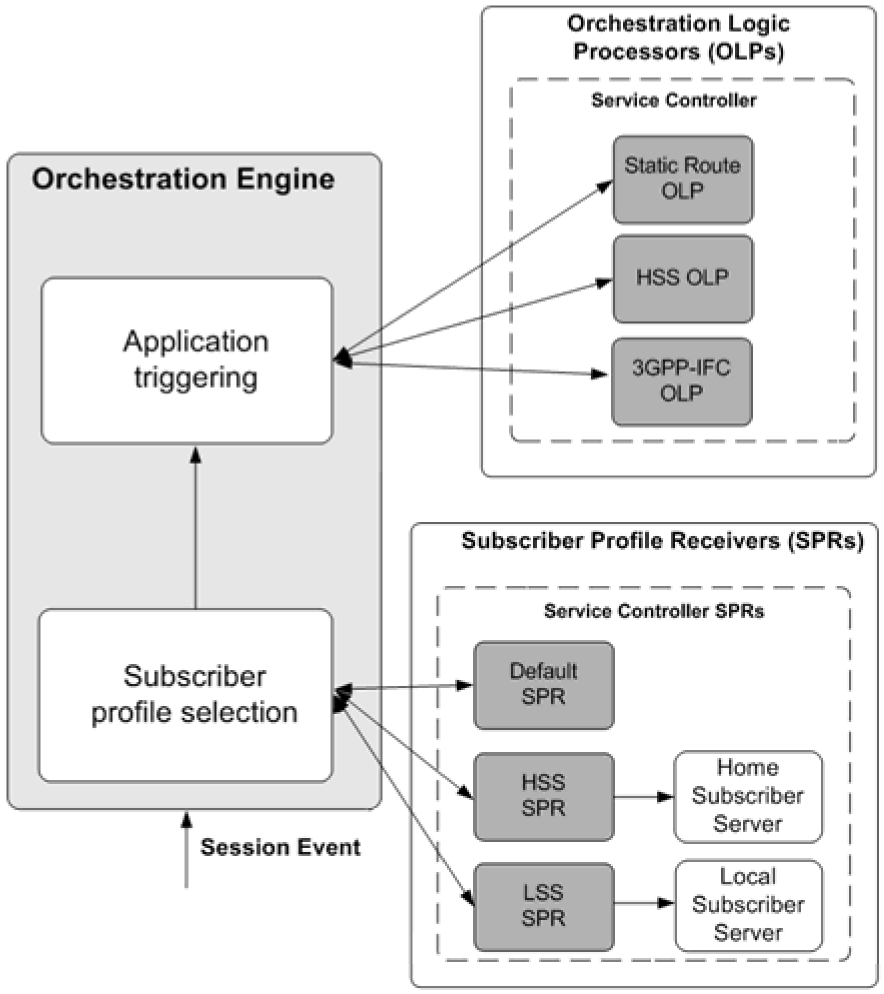
Table 1–1 describes components that the OE uses to retrieve a subscriber profile from a profile server and execute the orchestration logic.

Table 1–1 Service Controller 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.
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
Defined as a a pre configured list of applications	Default SPR	Static Route OLP

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

Note: You can add a new OPR to Service Controller, to connect to other profile sources that exist in the operator's network. Service Controller 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 Controller 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.
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 to 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 *Service Controller 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.
Enable SDR	BOOL	Specifies whether or not the OE generates SDRs. Possible values: ■ True ■ False Default value: True
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*@example.com matches sip:78880@example.com and sip:788801@example.com. ■ period (.), which matches one occurrence of any character. For example, sip:78880.0@example.com matches sip:7888010@example.com and sip:7888020@example.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!@example.com matches sip:subscriber10@example.com and sip:subscriber11@example.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 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 Controller monitoring in *Service Controller 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 known to Service Controller.

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 *Service Controller Modules Configuration Guide*.

3

Invoking Applications Based on the Previous Session Route

This chapter describes how to set up an application to run based on the previous route of the session in Oracle Communications Service Controller.

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 Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

Tagging a Session

To tag a session, edit the session XML source code. Add tags to the application header fields.

Invoking Applications Based on the Previous Session Route 3-1

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
- 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 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.

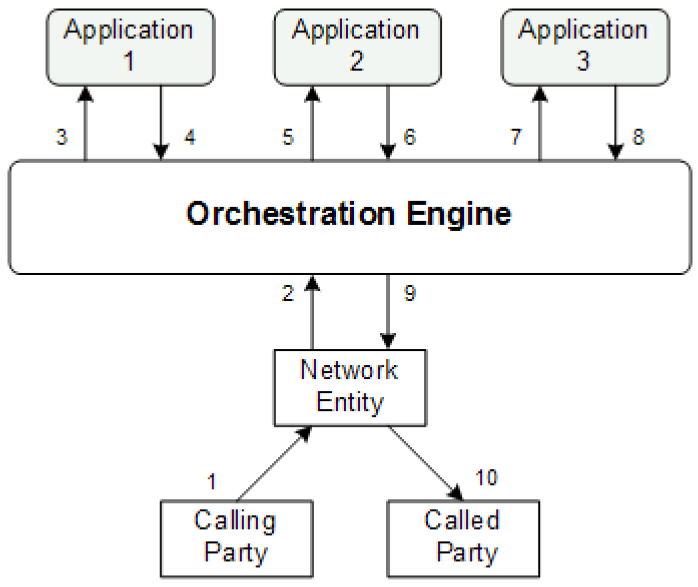
4 Defining the Orchestration Order of Messages Sent by a Called Party

This chapter describes how to define the order of messages sent by a called party for Oracle Communications Service Controller.

About the Orchestration Order

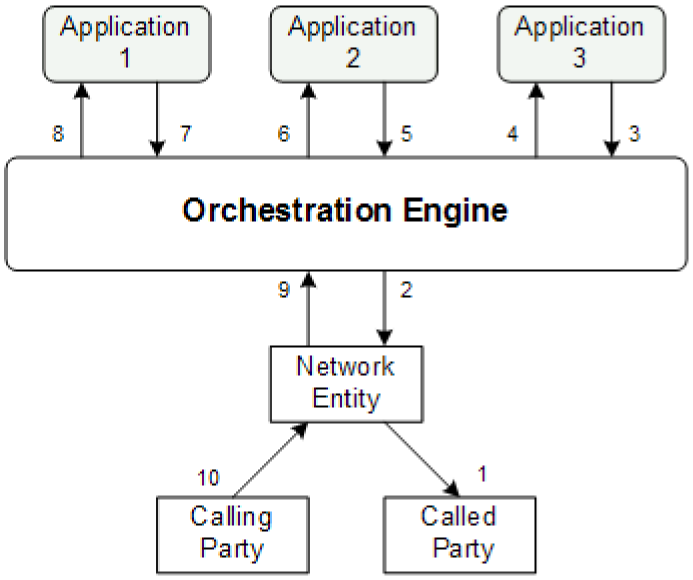
Orchestration logic defines how the OE routes messages generated by the calling party. For example, you can set an initial INVITE to be routed from Application 1 to Application 2 to Application 3. Figure 4–1 shows the order in which the OE routes an INVITE message from a calling party to a called party.

Figure 4–1 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 4–2 shows the order in which the OE routes an OK response from a called party to a calling party.

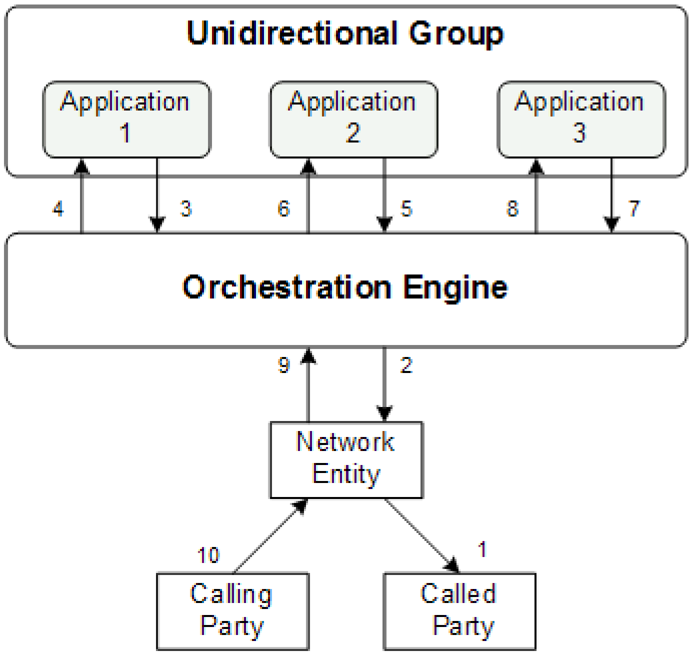
Figure 4–2 Routing an OK Message from a Called Party



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 4–3 shows how the OE routes a message generated by a called party through applications in a unidirectional group.

Figure 4–3 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 4–3, 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 Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

5

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

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

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.

Edit the XML source code for the OE session. To continue the session, enter a handling value of **0**. To end the session, enter a handling value of **1**.

Note: Service Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

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).

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

Note: Service Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

A Use Cases

This appendix presents some typical use cases that can be used as examples when creating orchestration logic flows for Oracle Communications Service Controller.

About the Use Cases

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

The uses cases in this appendix are demonstrated in their XML format. Use the XML as a starting point for your own orchestration flows.

Service Orchestration

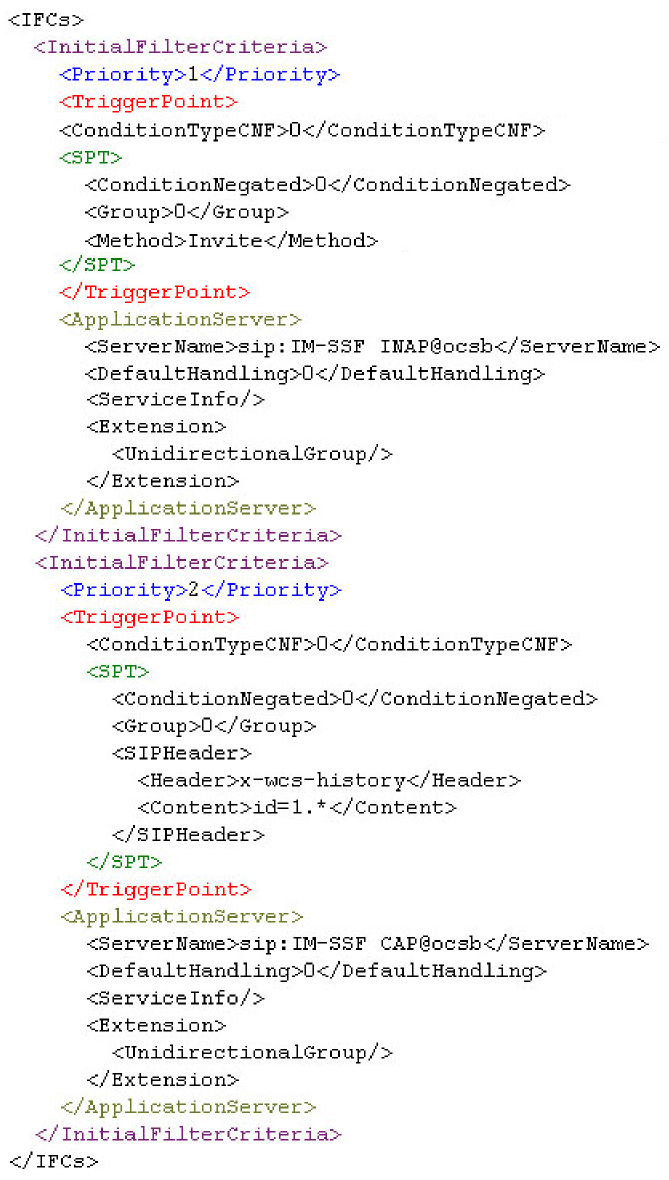
The following flows illustrate Service Controller orchestration capabilities.

IN Service Interaction

Figure A–1 shows a use case for how the Orchestration Engine forwards a session to an online charging application server and then to a VPN service.

Use the code sample below as a starting point to create your own logic flow for an IN Service Interaction.

Figure A–1 IN Service Interaction Source Code



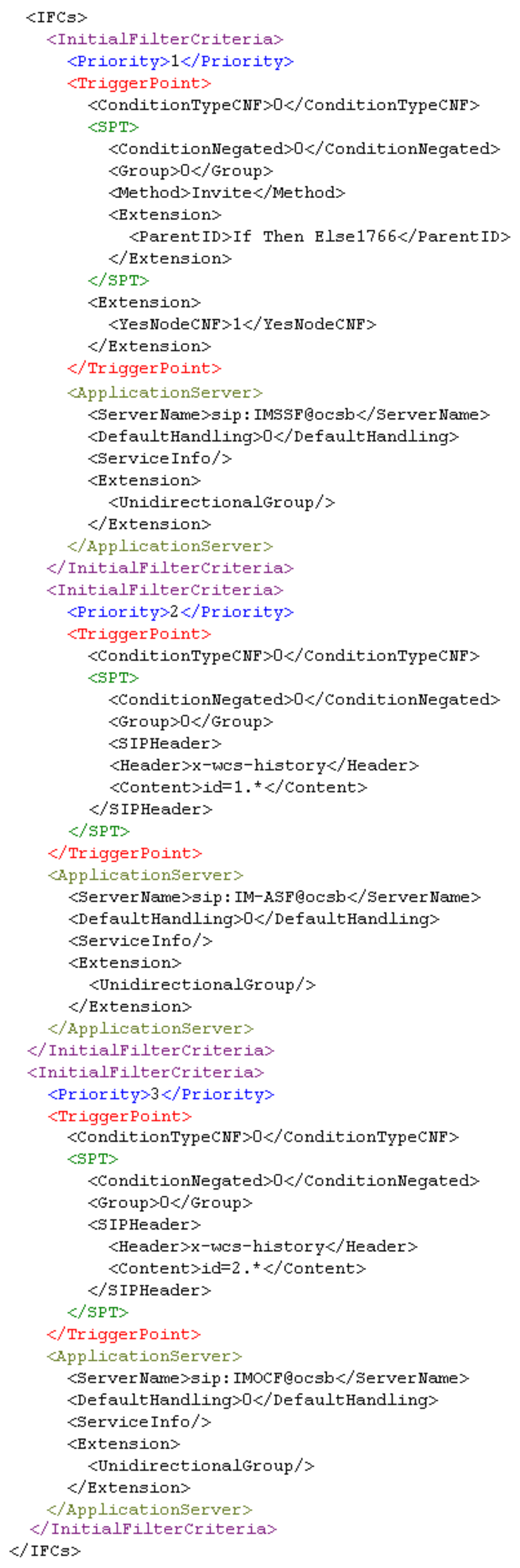
```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
    <ConditionTypeCNF>0</ConditionTypeCNF>
    <SPT>
      <ConditionNegated>0</ConditionNegated>
      <Group>0</Group>
      <Method>Invite</Method>
    </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:IM-SSF 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>
```

IMS Service Interaction

Figure A–2 shows a use case for how Service Controller communicates with the IMS network and provides service interaction based on the logic retrieved from the database.

Use the code sample below as a starting point to create your own logic flow for an IMS Service Interaction.

Figure A–2 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>
```

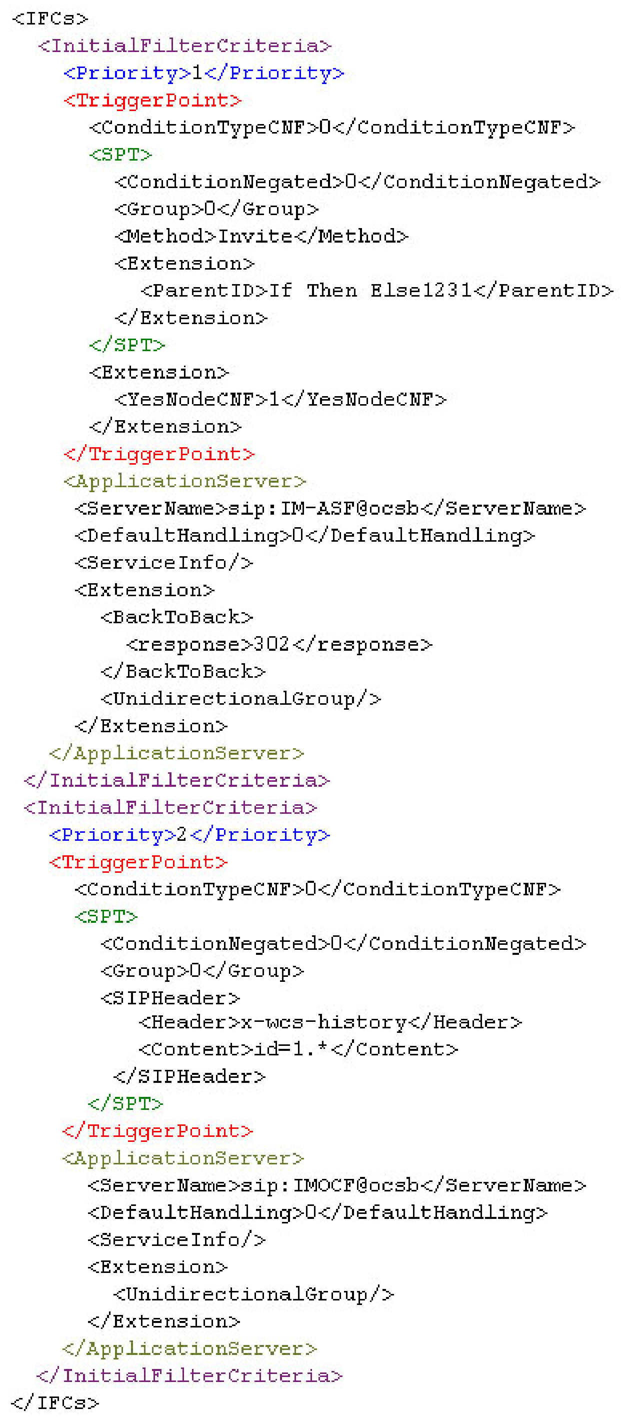
Forcing Back to Back

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.

Figure A–3 shows a use case for how 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.

Use the code sample below as a starting point to create your own B2B logic flow.

Figure A–3 Forced Back to Back Source Code



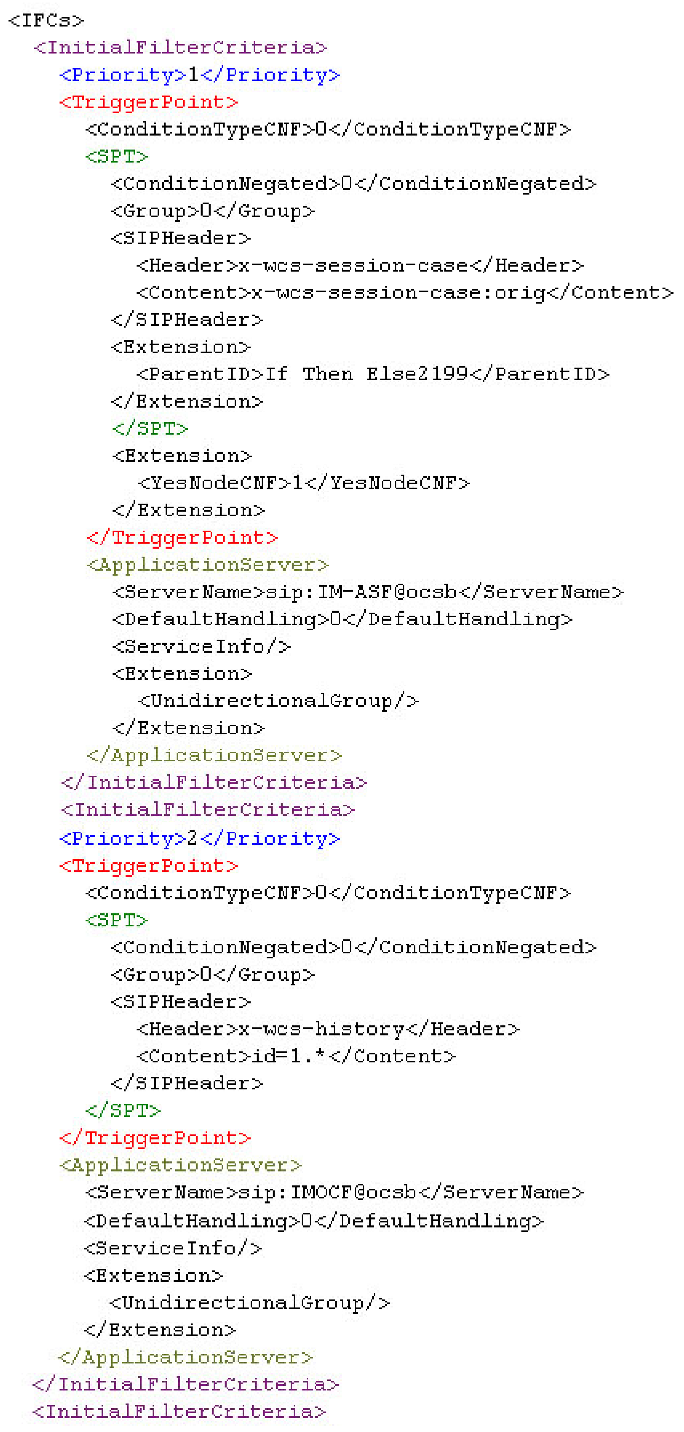
```
<IFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>Invite</Method>
        <Extension>
          <ParentID>If Then Else1231</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>
```

Choosing between Two Execution Paths

Figure A–4 and Figure A–5 show a use case for how, 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.

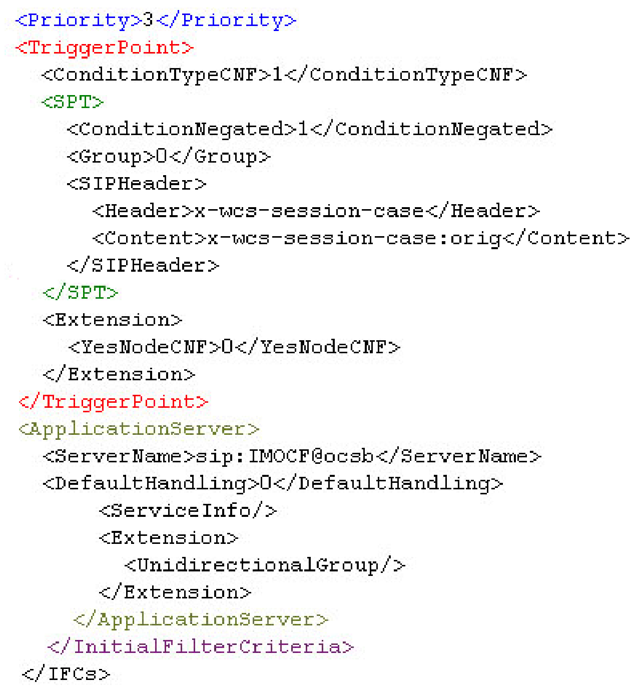
Use the code sample below as a starting point to create your own logic flow for continuing a session when conditions are not met.

Figure A–4 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–5 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>
```

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 Controller Orchestration Engine (OE).

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.

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.

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.

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).

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@example.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@example.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@example.com</RequestURI>
      </SPT>

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

    </TriggerPoint>

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

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@example.com</ServerName>
    <DefaultHandling>0</DefaultHandling>
  </ApplicationServer>
```

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

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.

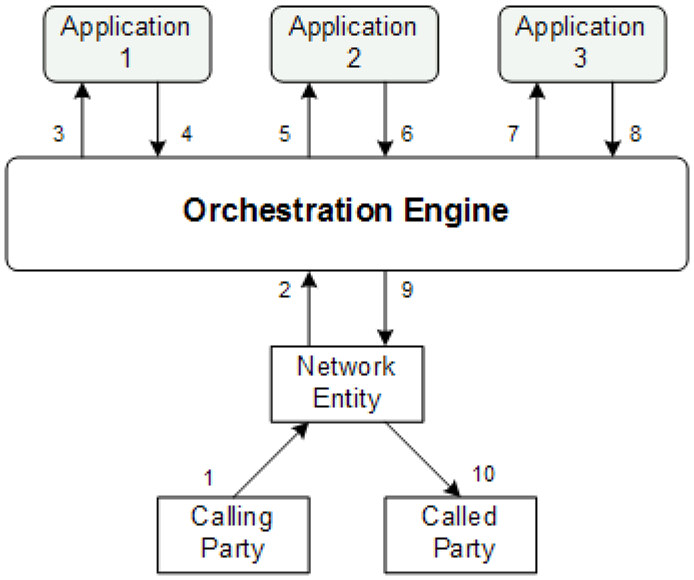
The higher the rule's priority number, the lower priority the filter criterion has. This means that a filter criterion with a higher value of priority number is assessed after the filter criteria with a smaller priority number has been assessed. 0 (zero) means the highest priority. 100 means the lowest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

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

Specifying the Order of Message Routing

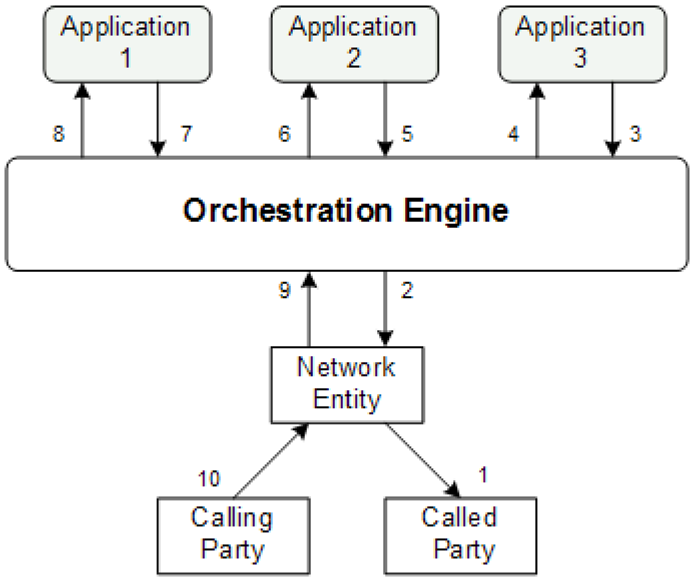
The iFC defines how the OE routes messages generated by the calling party. For example, you can set an initial INVITE to be routed from Application 1 to Application 2 to Application 3. Figure B–1 shows the order in which the OE routes an INVITE message from a calling party to a called party.

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.

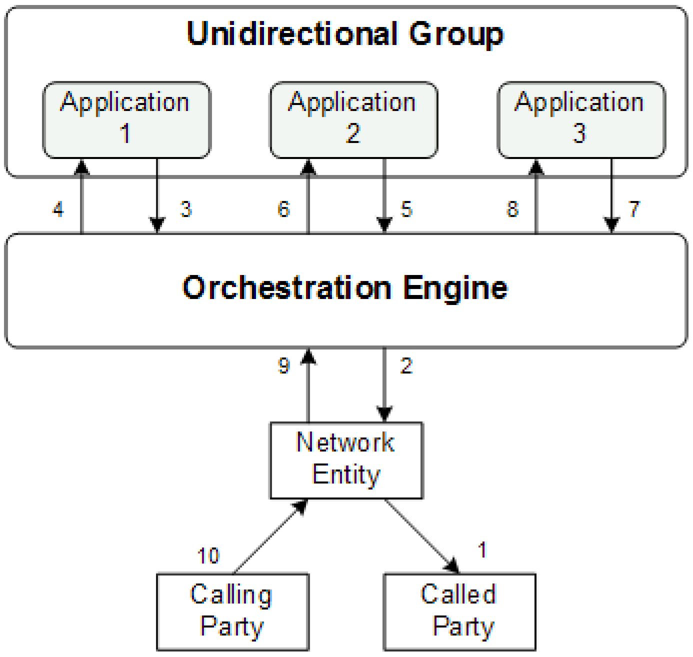
Figure B–2 Routing an OK Message from a Called Party



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 Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

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>
```

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 Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

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).

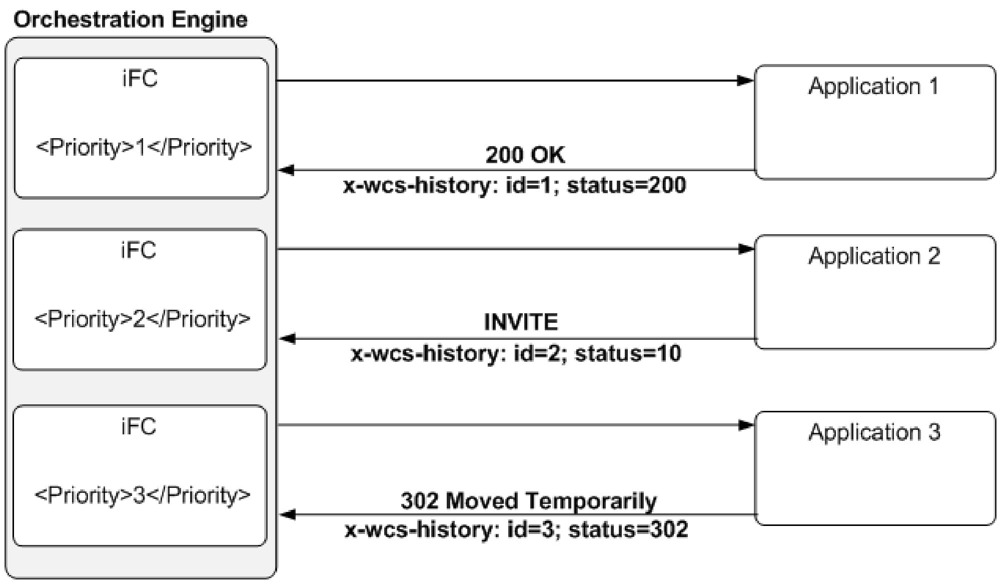
Figure B–4 shows a scenario in which the OE routes the session as follows:

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



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.

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>
```

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>
```

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 Controller 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 *Service Broker Online Mediation Controller Implementation Guide Release 6.1*.

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
```

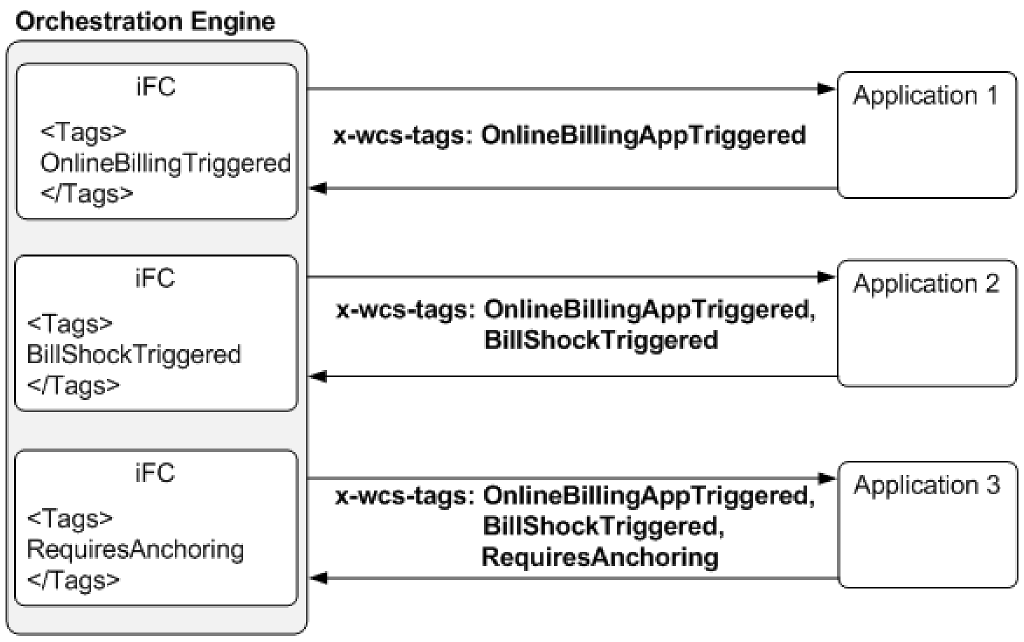
The application returns the message to the OE with the **x-wcs-tags** header intact. As the message passes through applications in the orchestration chain, the **x-wcs-tags** header accumulates the contents of all `<Tags>` elements defined for applications in the chain.

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


Orchestration Engine
iFC
<Tags>
OnlineBillingTriggered
</Tags>
iFC
<Tags>
BillShockTriggered
</Tags>
iFC
<Tags>
RequiresAnchoring
</Tags>
x-wcs-tags: OnlineBillingAppTriggered
x-wcs-tags: OnlineBillingAppTriggered, BillShockTriggered
x-wcs-tags: OnlineBillingAppTriggered, BillShockTriggered, RequiresAnchoring
Application 1
Application 2
Application 3

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.

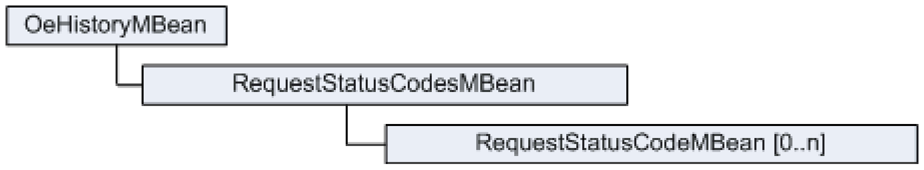
```
<iFCs>
  <InitialFilterCriteria>
    <Priority>1</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
      </SPT>
    </TriggerPoint>
  <ApplicationServer>
 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