

Oracle® Communications Service Broker

SIP Developer's Guide for GSM

Release 6.0

E23533-01

March 2012

Copyright © 2010, 2012, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	ix
Audience	ix
Documentation Accessibility	ix
Related Documents	ix
Downloading Oracle Communications Documentation	x
1 Principles of the Oracle Communications Service Broker NG-IN Solution	
Introduction	1-1
Solution Architecture	1-2
Session Control	1-2
Charging	1-2
User Interaction	1-3
Multleg Control	1-3
Information Exchange through the SIP Interface	1-3
Information Exchange using SIP Headers	1-3
Information Exchange using SIP Body	1-4
Setting the Content-Type Header for BER Tunneling	1-4
2 Developing a SIP Call Control Application	
Controlling a Call	2-1
Invoking a SIP Application	2-1
Exposing Nature of Address	2-4
Exposing Diversion Information	2-5
Developing an Initial Call Control Application	2-6
Updating the Called Party Number	2-7
Creating CAP Connect	2-8
Leaving the Called Party Number Unmodified	2-9
Updating the Calling Party Number	2-10
Developing a Full Call Control Application	2-11
Handling the SDP	2-12
Handling the SIP Route Header	2-13
Updating the Called Party Number	2-13
Leaving the Called Party Number Unmodified	2-17
Updating the Calling Party Number	2-19
Updating the Nature of Address	2-20

Controlling the EDPs Arming	2-21
Receiving Call Events Notifications	2-22
Terminating a Call	2-28
Service Broker Error Responses	2-29
Rejecting a Call	2-29
Controlling the CAP Release Cause	2-30

3 Developing a Charging Application

About CAP Charging Operations	3-1
Performing FurnishChargingInformation Operation	3-1
Performing FurnishChargingInformation in an Application Initiated Call.....	3-4
Performing SendChargingInformation Operation	3-5
Performing ApplyCharging Operation	3-5
Receiving a Report	3-5
Performing CallInformationRequest Operation	3-8
Receiving a Report	3-9

4 Developing a SIP User Interaction Application

Providing User Interaction Functionality	4-1
About the Media Resource Table.....	4-2
Suspending Call Processing for Pre-Call Announcements	4-4
Using an External Media Resource	4-6
Playing an Announcement	4-7
Terminating a User Interaction Session.....	4-9
Playing an Announcement and Collecting Information	4-12
Using an Internal Media Resource	4-17
Playing an Announcement	4-18
Playing an Announcement and Collecting Information	4-22
SIP MRF Integration for User Interaction Functionality	4-27

5 Developing a Multi-Leg Call Control Application

Providing Multi-Leg Call Control	5-1
About CAP Call Leg Representation.....	5-1
Creating a New Call Leg	5-2
Creating a New Leg in a New Call.....	5-3
Creating a New Leg in an Existing Call.....	5-4
Restricting Calling Line Identification for a Calling Party Number	5-4
Restricting Calling Line Identification on a Permanent Basis	5-4
Restricting Calling Line Identification on a Per-Message Basis	5-4
Disconnecting a Call Leg	5-5
Removing a Leg from a Call	5-5
Connecting Call Legs	5-6
Multi-Leg Control Example	5-6

6 Understanding the Service Broker NG-IN Solution for Presence and Subscriber Status Applications

Introduction.....	6-1
Solution Architecture	6-1
SIP SUBSCRIBE and SIP NOTIFY Interface.....	6-2
Exchanging Information Through the SIP Interface.....	6-3

7 Developing a Presence and Subscriber Status SIP Application

Understanding Common SIP Interface Concepts.....	7-1
Specifying the Address of an SS7 Entity.....	7-1
Specifying the Identity of a Mobile Subscriber.....	7-3
SIP NOTIFY Message Body Formats.....	7-3
Specifying Supported SIP NOTIFY Message Body Formats	7-4
Setting the Expires Header	7-4
Handling SIP Errors.....	7-4
Obtaining Subscriber's State and Location.....	7-5
Generating a SIP SUBSCRIBE Message	7-5
Common SIP Headers	7-5
Event Header	7-5
Processing a SIP NOTIFY Request.....	7-5
Subscription-State Header	7-6
Content-Type Header.....	7-6
SIP Message Body.....	7-7
Obtaining Mobile Subscriber's Subscription Information	7-8
Generating a SIP SUBSCRIBE Message	7-8
Common SIP Headers	7-8
Event Header	7-9
Content-Type Header	7-9
Processing the SIP NOTIFY Message.....	7-9
Subscription-State Header	7-9
Modifying Mobile Subscriber's Information.....	7-10
Generating a SIP Subscribe Message.....	7-10
Common SIP Headers	7-11
Event Header	7-11
Content-Type Header.....	7-11
Processing a SIP Notify Message.....	7-11
Subscription-State Header	7-11
Content-Type Header.....	7-12

8 Developing a Presence and Subscriber Status SIP Application

Understanding Common SIP Interface Concepts.....	8-1
Specifying the Address of an SS7 Entity.....	8-1
Specifying the Identity of a Mobile Subscriber.....	8-3
SIP NOTIFY Message Body Formats.....	8-3
Specifying Supported SIP NOTIFY Message Body Formats	8-4
Setting the Expires Header	8-4

Handling SIP Errors.....	8-4
Obtaining Subscriber's State and Location.....	8-5
Generating a SIP SUBSCRIBE Message	8-5
Common SIP Headers	8-5
Event Header	8-5
Processing a SIP NOTIFY Request.....	8-6
Subscription-State Header	8-6
Content-Type Header.....	8-6
SIP Message Body	8-7
Obtaining Mobile Subscriber's Subscription Information	8-8
Generating a SIP SUBSCRIBE Message	8-8
Common SIP Headers	8-8
Event Header	8-9
Content-Type Header	8-9
Processing the SIP NOTIFY Message.....	8-9
Subscription-State Header	8-9
Modifying Mobile Subscriber's Information.....	8-10
Generating a SIP Subscribe Message.....	8-10
Common SIP Headers	8-11
Event Header	8-11
Content-Type Header.....	8-11
Processing a SIP Notify Message	8-11
Subscription-State Header	8-11
Content-Type Header.....	8-12

9 Developing a Short Message Application

About Solution Architecture.....	9-1
Sending a Short Message from an Application to an SMSC.....	9-2
About Methods of Specifying Short Message Parameters.....	9-3
Specifying the Identity of an Application.....	9-3
Specifying the Identity of a Mobile Subscriber	9-3
Specifying the Message Content Type	9-4
Setting the Validity Period of a Message.....	9-4
Triggering an SMSC to Send a Predefined Short Message to a Mobile Subscriber.....	9-4
Specifying the Message Mode, Type, and Privacy Indicator.....	9-5
Specifying the Message Priority.....	9-5
Requesting a Delivery Receipt or Acknowledgement.....	9-5
Tunneling an SMPP Message using the XER or BER Format.....	9-5
Processing Headers of a Delivery Receipt or Acknowledgement	9-6
Receiving a Short Message Sent by a Mobile Subscriber.....	9-6
Processing Headings of a Short Message Sent by a Mobile Subscriber	9-7
Handling Errors	9-7

A Detailed Sequence Diagram Examples

Personalized Ring Back Tone	A-1
Message Dumps	A-4
Message #1: CAP InitialDP.....	A-5

Message #2: SIP INVITE	A-6
Message #3: SIP_INVITE	A-8
Message #4: CAP InitiateCallAttempt.....	A-9
Message #5: CAP InitiateCallAttempt Result.....	A-10
Message #6: SIP 183 SESSION PROGRESS	A-11
Message #7: CAP RequestReportBCSMEvent.....	A-12
Message #8: CAP ContinueWithArgument.....	A-13
Message #9: SIP INVITE	A-14
Message #10: CAP InitiateCallAttempt.....	A-15
Message #11: CAP InitiateCallAttempt Result.....	A-16
Message #12: SIP 183 SESSION PROGRESS	A-17
Message #13: CAP RequestReportBCSMEvent.....	A-18
Message #14: CAP ContinueWithArgument.....	A-19
Message #15: CAP EventReportBCSM.....	A-20
Message #16: SIP 180 RINGING	A-21
Message #17: CAP EventReportBCSM.....	A-22
Message #18: SIP 180 RINGING	A-23
Message #19: SIP 183 SESSION PROGRESS	A-24
Message #20: CAP DisconnectLeg.....	A-25
Message #21: CAP DisconnectLeg Result.....	A-26
Message #22: CAP RequestReportBCSM	A-27
Message #23: CAP ContinueWithArgument.....	A-28
Message #24: CAP MoveLeg.....	A-29
Message #25: CAP MoveLeg Result.....	A-30
Message #26: CAP ContinueWithArgument.....	A-31
Message #27: CAP EventReportBCSM.....	A-32
Message #28: SIP 200 OK	A-33
Message #29: SIP 200 OK	A-34
Message #30: CAP ContinueWithArgument.....	A-35
Message #31: CAP EventReportBCSM.....	A-36
Message #32: SIP 200 OK	A-37
Message #33: CAP ContinueWithArgument.....	A-38
Message #34: SIP BYE.....	A-39
Message #35: SIP 200 OK	A-40
Message #36: CAP DisconnectLeg.....	A-41
Message #37: CAP DisconnectLeg Result.....	A-42
Message #38: CAP ContinueWithArgument.....	A-43
Message #39: SIP reINVITE	A-44
Message #40: CAP MoveLeg.....	A-45
Message #41: CAP MoveLeg Result.....	A-46
Message #42: CAP ContinueWithArgument.....	A-47
Message #43: SIP 200 OK	A-48
Message #44: CAP EventReportBCSM.....	A-49
Message #45: SIP BYE.....	A-50
Message #46: SIP 200 OK	A-51
Message #47: SIP BYE.....	A-52
Message #48: SIP 200 OK	A-53

Message #49: CAP ReleaseCall A-54

Preface

This document provides detailed guidances for developing SIP applications and integrating these applications with the Oracle Communications Service Broker Next Generation - Intelligent Network (NG-IN) solution.

Audience

This document is intended for developers who want to develop SIP applications and integrate these applications with the Service Broker NG-IN solution.

This document assumes that the reader is already familiar with the following:

- Service Broker architecture and concepts
- Session Initiation Protocol (SIP) and the SIP-specific event notification
- CAP protocol specifications and procedures
- MAP protocol specifications and procedures

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.

Related Documents

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

Oracle Communications Service Broker User Manuals

- *Oracle Communications Service Broker Release 6.0 Concepts Guide*
- *Oracle Communications Service Broker Release 6.0 Online Mediation Controller Implementation Guide*

- *Oracle Communications Service Broker Release 6.0 Signaling Domain Configuration Guide*
- *Oracle Communications Service Broker Release 6.0 Processing Domain Configuration Guide*

SIP Standards

- IETF RFC 3261, Session Initiation Protocol
- IETF RFC 3323, A Privacy Mechanism for the Session Initiation Protocol (SIP)
- IETF RFC 3325, Private Extensions to the Session Initiation Protocol
- IETF Internet Draft, Diversion Indication in SIP, August 25, 2004
- IETF Internet Draft, A Header to Deliver the Calling Party Category, October 21, 2005

Mobile Application Part (MAP) Standards

- ETSI TS 129 002 (3GPP TS 29.002) version 7.10.0, Mobile Application Part (MAP) specification
- ETSI TS 123 078 (3GPP TS 23.078) version 7.9.0, Customized Applications for Mobile network, Enhanced Logic (CAMEL) Phase X; Stage 2

Encoding Standards

- IETF RFC 3863, Presence Information Data Format (PIDF)
- ITU-T X.693, ASN.1 Encoding Rules, XML Encoding Rules (XER)

Downloading Oracle Communications Documentation

Oracle Communications Service Broker documentation is available from the Oracle Software Delivery Web site:

<http://edelivery.oracle.com/>

Additional Oracle Communication documentation is available from Oracle Technology Network:

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

Principles of the Oracle Communications Service Broker NG-IN Solution

This chapter describes the principles of the Oracle Communications Service Broker NG-IN solution.

Introduction

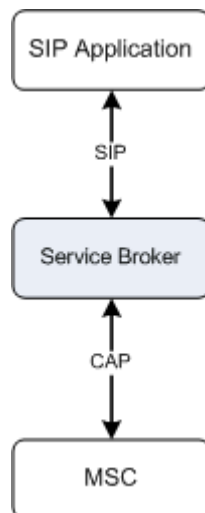
Service Broker NG-IN solution enables a SIP application to control an IN-enabled MSC or SSP in a legacy network.

With a Service Broker NG-IN solution, you can:

- Deploy new SIP applications and deliver them towards a legacy network
- Migrate legacy SCP-based applications to SIP-based applications and deliver them towards a legacy network.

Figure 1-1 shows a SIP application that controls an IN-enabled MSC in a legacy network.

Figure 1-1 Architecture for Controlling a Legacy IN-Enabled MSC by a SIP Application



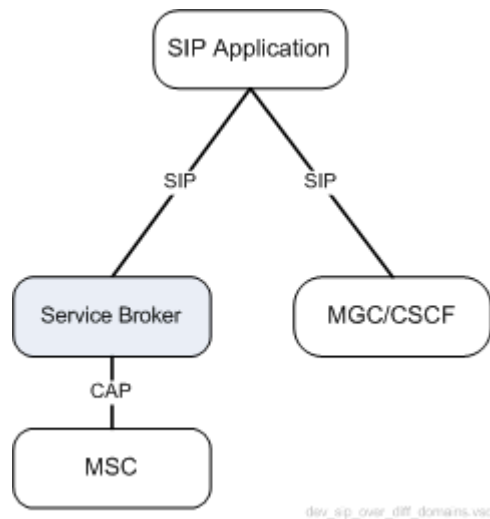
Service Broker provides SIP applications with a standard SIP interface to control IN-enabled MSCs. This enables a SIP application to control an MSC in a legacy network as it controls an MGC or CSCF in a SIP or IMS network. Furthermore, from

the application developer's perspective, the application's control over an MSC does not require any network-specific customization.

Service Broker enables advanced applications (for example charging applications) to control IN-specific parameters by exposing these parameters towards the application through the SIP interface.

Figure 1–2 shows a SIP application that provides call control to an MGC or CSCF in a SIP or IMS network, and to an IN-enabled MSC in a legacy network.

Figure 1–2 Architecture for controlling an MSC and MGC/CSCF by a SIP Application



Solution Architecture

The Service Broker NG-IN solution is composed of the following components:

- One or more SIP applications
- Service Broker

In an NG-IN solution, Service Broker has two external interfaces:

- Northbound SIP interface towards SIP applications
- Southbound IN interface towards the MSC

Session Control

Service Broker enables a SIP application to control an IN call through the SIP interface. An application may operate in a full call control mode or an initial call control mode acting as a SIP B2BUA or as a SIP Redirect Server accordingly.

Charging

Service Broker enables a SIP application to control an MSC for online and offline charging services. Charging operations are transferred from the application to Service Broker using SIP INFO messages. These messages carry an XML representation of the charging operation that needs to be performed.

For example, an application may send a SIP INFO message with a body that carries an XML representation of a CAP phase 4 FurnishChargingInformation operation. Upon

receiving a SIP INFO, Service Broker sends a CAP FurnishChargingInformation towards the MSC.

User Interaction

Service Broker enables a SIP application to interact with a call party for providing service announcement to the call party with or without DTMF collection.

Based on application's instructions, Service Broker uses different media resources:

- MSC's internal resource
- External resource, that is gsmSRF
- MRF

When Service Broker uses internal or external resources, user interaction operations are transferred from the application to Service Broker using SIP INFO messages that carry an XML representation of the user interaction operation to be performed.

For example, an application may send a SIP INFO message with a body that carries the XML representation of CAP phase 4 PlayAnnouncement operation.

Multileg Control

Service Broker enables a SIP application to control individual parties in a call. For example, an application may create a new leg in an existing call or in a new call, connect two or more legs, split a leg out from the call, and more.

Multi-leg control is used by an application acting as a B2BUA to provide enhanced services, such as personalized ringback tone and click-to-dial.

Information Exchange through the SIP Interface

Service Broker exchanges information with the SIP application through the common SIP interface using two different mechanisms:

- Using SIP headers
- Using SIP message body

The following sections describe these two mechanisms.

Information Exchange using SIP Headers

To provide the application with the call related information received through the CAP interface, Service Broker uses the headers of the messages sent to the application.

For example, when Service Broker receives an InitialDP operation through the CAP interface, Service Broker sends a SIP INVITE message to the application and sets the Request-URI to the called party address as received in the CAP InitialDP operation. In the other direction, Service Broker uses the headers of the messages received from the application to construct CAP operation and send it towards the MSC.

In addition, to exchange information with a SIP application, Service Broker uses SIP tokens. For example Service Broker uses the **noa** token to exchange the nature of address information of various call parties with the SIP application.

Information Exchange using SIP Body

Service Broker uses the SIP message body to exchange two types of information:

- IN parameters, which are not naturally transferred using the SIP headers. For example, Service Broker may use the SIP INVITE message body to propagate the IN BearerCapability parameter towards the application.

IN parameters are exchanged using the SIP body in both directions: from Service Broker to the application and from the application to Service Broker.

- SDP, which contains call leg information.

Setting the Content-Type Header for BER Tunneling

When Service Broker tunnels BER through the SIP interface, Service Broker uses the "op" token and "dir" token of the SIP Content-Type header to describe the message that the body contains. [Table 1-1](#) describes these tokens.

Table 1-1 Content-Type Tokens

Token	Description
op	Specifies the code of the tunneled operation
dir	Specifies a direction of the tunneled operation, that is whether the operation is an invocation or result. Possible values: <ul style="list-style-type: none"> ■ Invoke ■ Result

For example:

```
Content-Type: application/cap-phase4+ber;op=123;dir=invoke
```

```
Content-Type: application/cap-phase4+ber;op=456;dir=result
```

Developing a SIP Call Control Application

This chapter describes how to develop a SIP call control application.

Controlling a Call

When Service Broker invokes a call control application, the application can perform one of the following actions:

- Rejecting the call
- Allowing the call to continue leaving the call information unmodified
- Allowing the call to continue with the call information modified. The application performs this action by providing Service Broker with the call routing information. When Service Broker receives the call routing information, Service Broker propagates this information towards the MSC through the IN interface.

If the application decides to allow the call to continue, the application can do this in one of the following forms:

- Routing the call while retaining call control for the entire call duration. An application that retains call control for the entire call duration is known as a full call control application.
- Routing the call without retaining call control for the entire call duration. An application that routes the call to destination without retaining call control for the entire call duration is known as initial call control application.

Invoking a SIP Application

Table 2–1 Invoking a SIP Application: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

Acting as a standard SIP entity, Service Broker invokes a SIP application by sending a SIP INVITE message. The Service Broker sets the SIP INVITE content based on the information received in the CAP InitialDP operation.

Table 2–2 shows the content of the SIP INVITE message as set by Service Broker. The SIP Header column describes the SIP INVITE headers. The Source column includes the source of the information used by Service Broker to set the SIP INVITE message.

Table 2–2 Service Broker Mapping of CAP 4 InitialDP Operation to SIP INVITE Message

SIP Header	Source
Request-URI :: User part (see the note for Request-URI below the table)	InitialDP :: CalledPartyNumber OR InitialDP :: CalledPartyBCDNumber CalledPartyBCDNumber is used only when CalledPartyNumber is not included in InitialDP.
Request-URI :: Domain part	Service Broker configuration
To :: User part	Equals to the value set in RequestURI :: User part
To :: Domain part	Service Broker configuration
From :: User part	InitialDP :: AdditionalCallingPartyNumber If AdditionalCallingPartyNumber is not included in InitialDP, the From header is set from InitialDP :: CallingPartyNumber. If InitialDP :: CallingPartyNumber :: Address presentation restricted indicator is set to "presentation restricted": The From header is set as follows: From: "Anonymous" sip:anonymous@anonymous.invalid The Privacy header is set as follows: Privacy: id
From :: Domain part	Service Broker configuration
P-Asserted-Identity :: User part (see the note for P-Asserted-Identity below the table)	InitialDP :: CallingPartyNumber
P-Asserted-Identity :: Domain part	Service Broker configuration
Privacy	The Privacy header is set as follows: Privacy:id The Privacy header is included only if InitialDP :: CallingPartyNumber :: Address presentation restricted indicator is set to "presentation restricted".
Diversion - top most header :: name-addr	InitialDP :: RedirectingPartyID
Diversion - top most header :: Counter	InitialDP :: RedirectionInformation :: RedirectionCounter
Diversion - top most header :: Reason	InitialDP :: RedirectionInformation :: RedirectingReason
Diversion - top most header :: Domain part	Service Broker configuration
Diversion - bottom most header :: name-addr (see the note for Diversion - bottom most header below the table)	InitialDP :: OriginalCalledPartyID
Diversion - bottom most header :: Counter	InitialDP :: RedirectionInformation :: RedirectionCounter

Table 2–2 (Cont.) Service Broker Mapping of CAP 4 InitialDP Operation to SIP INVITE

SIP Header	Source
Diversion - bottom most header :: Reason	InitialDP :: RedirectionInformation :: OriginalRedirectionReason
Diversion - bottom most header :: Domain part	Service Broker configuration
Route	<p>The "orig" token is included if InitialDP :: eventTypeBCSM set to CollectedInfo or to AnalyzedInformation.</p> <p>The "term" token is included if InitialDP :: EventTypeBCSM set to TermAttemptAuthorized.</p> <p>Service Broker sets the Route header with a token that indicates call direction, that is originating call or terminating call. For originating call, Service Broker sets the token to "orig" while for terminating call token is set to "term".</p> <p>The SIP INVITE received by application may include several Route headers.</p>
x-wcs-session-case	<p>If InitialDP :: EventTypeBCSM set to CollectedInfo or to AnalyzedInformation, the x-wcs-session-case is set as follows:</p> <p>x-wcs-session-case:orig</p> <p>If InitialDP :: EventTypeBCSM set to TermAttemptAuthorized, the x-wcs-session-case is set as follows:</p> <p>x-wcs-session-case:term.</p>
CPC	<p>InitialDP :: CallingPartysCategory</p> <p>If InitialDP :: CallingPartysCategory is one of the following:</p> <ul style="list-style-type: none"> ▪ Operator, Language French ▪ Operator, Language English ▪ Operator, Language German ▪ Operator, Language Russian ▪ Operator, Language Spanish <p>the CPC header is set to "operator" and the specific language is set in the SIP Accept-Language header</p>
Accept-Language	<p>InitialDP :: CallingPartysCategory</p> <p>The SIP Accept-Language is included only if InitialDP :: CallingPartysCategory is set to one of the following:</p> <ul style="list-style-type: none"> ▪ Operator, Language French ▪ Operator, Language English ▪ Operator, Language German ▪ Operator, Language Russian ▪ Operator, Language Spanish <p>In this case, SIP Accept-Language is set to the language (for example, French).</p>
P-Charging vector :: icid-value	<p>InitialDP :: CallReferenceNumber</p> <p>If CallReferenceNumber is not included in the InitialDP, the P-Charging-Vector header is not included in the SIP INVITE.</p>

Table 2–2 (Cont.) Service Broker Mapping of CAP 4 InitialDP Operation to SIP INVITE

SIP Header	Source
P-Charging vector :: icid-gen-addr	IM_SCF instance name
P-Charging vector :: Orig-ioi	Not set
P-Charging vector :: Term-ioi	Not set
Subject	The Subject header is set as follows: Subject:call control
x-wcs-mobile-number	InitialDP :: IMSI
x-wcs-service-key	InitialDP :: ServiceKey
x-wcs-network-name	Service Broker configuration The x-wcs-network-name enables the Service Broker to provide the application with enhanced information about the underlying network. This information can be used by application to apply specific logic based on the network where call was initiated.
x-wcs-msc-address	InitialDP :: MscAddress

Notes to Table 2-2:

- Request-URI

Service Broker sets the Request-URI with a **noa** token. For more information on the noa token, see "[Exposing Nature of Address](#)".

- P-Asserted-Identity

The Service Broker sets the P-Asserted-Identity with a noa token. For more information on the noa token, see "[Exposing Nature of Address](#)".

- Diversion - bottom most header

For more information, see "[Exposing Diversion Information](#)".

- x-wcs headers

These are vendor specific headers used by Service Broker.

Exposing Nature of Address

Table 2–3 Exposing Nature of Address: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

Service Broker exposes CAP nature of address information towards the SIP application using a vendor specific token, named noa. The noa token carries the nature of address of various call parties as follows:

- Nature of address of the called party number provided by noa token in the Request-URI header of the SIP INVITE, which is sent to the application
- Nature of address of the calling party number provided by noa token in the P-Asserted-Identity header of the SIP INVITE, which is sent to the application

Table 2–4 describes how Service Broker sets the noa token for various nature of address values.

Table 2–4 NOA Token

Call party nature of address	NOA token content
subscriber number (national use)	subscriber
unknown (national use)	unknown
national (significant) number (national use)	national
International	noa token is not used

As shown in Table 2–4, Service Broker does not set the noa token for nature of address of type “International”. When the nature of address of one of the call parties is set to “International”, the user part in the corresponding SIP header is prefixed with “+”. For example:

- For the calling party nature of address of type “International”, Service Broker sets the P-Asserted-Identity in the SIP INVITE, which is sent to the application, as follows:

```
P-Asserted-Identity:
<sip:+97297888019@domain:5060>
```

- For the calling party nature of address of type “national (significant) number”, Service Broker sets the P-Asserted-Identity in the SIP INVITE, which is sent to the application, as follows:

```
P-Asserted-Identity:
<sip:97888019@domain:5060;noa=national>
```

Exposing Diversion Information

Table 2–5 Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

Service Broker uses the standard SIP Diversion header, as described in Steve Levy, J. R. Yang, "Diversion Indication in SIP draft-levy-sip-diversion-08", to provide CAP redirection information towards the SIP application. Service Broker sets the SIP INVITE message with one or more Diversion headers depending on availability of information in the CAP InitialDP operation as follows:

- CAP OriginalCalledPartyID is provided to the application using the bottom most SIP Diversion header (for more information, see Table 2–2).
- CAP RedirectingPartyID is provided to the application using the top most Diversion header (for more information, see Table 2–2).
- CAP RedirectionInformation is provided to the application as defined in Table 2–6.

Table 2–6 CAP RedirectionInformation

CAP RedirectionInformation Parameter: Value	Used SIP Header
OriginalRedirection Reason :: unknown/not available	bottom most diversion header, reason=unknown

Table 2–6 (Cont.) CAP RedirectionInformation

CAP RedirectionInformation Parameter: Value	Used SIP Header
OriginalRedirection Reason :: user busy (national use)	bottom most diversion header, reason=user-busy
OriginalRedirection Reason :: no reply (national use)	bottom most diversion header, reason=no-answer
OriginalRedirection Reason :: unconditional (national use)	bottom most diversion header, reason= unconditional
RedirectionCounter :: binary number between 1 and 5	diversion-counter
RedirectingReason :: unknown/not available	top most diversion header, reason=unknown
RedirectingReason :: user busy	top most diversion header, reason=user-busy
RedirectingReason :: no reply	top most diversion header, reason=no-answer
RedirectingReason :: unconditional	top most diversion header, reason= unconditional
RedirectingReason :: deflection during alerting	top most diversion header, reason= deflection
RedirectingReason :: deflection immediate response	top most diversion header, reason= follow-me
RedirectingReason :: mobile subscriber not reachable	top most diversion header, reason=out-of-service

Developing an Initial Call Control Application

Table 2–7 Developing an Initial Call Control Application: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To provide an initial call control, the application responds to the SIP INVITE message with a SIP 302 Moved Temporarily message.

An initial call control application can perform one of the following actions:

- Updating the called party number (that is to replace the number dialed by the calling party with a new number)
- Leaving the called party number unmodified
- Updating the calling party number

The following sections describe how to implement these two options.

Note: An initial call control application may also reject a call. This functionality is described in "[Rejecting a Call](#)".

Updating the Called Party Number

Table 2–8 *Updating the Called Party Number: Applicable CAP Phases*

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To update the called party number (that is to replace the number dialed by the calling party with a new number), the application sets the SIP 302 Moved Temporarily to the new destination address.

The new destination address is set in the user part of the Contact header. This makes Service Broker to respond to the CAP InitialDP with a CAP Connect operation.

Note: The application can set the user part with digits only.

Figure 2–1 shows the high level architecture for an initial call control application that updates the called party number.

Figure 2–1 *Architecture for Updating the Called Party Number by an Initial Call Control Application over a SIP Network*

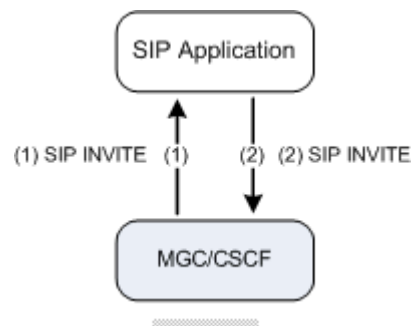


Figure 2–2 shows the same application as shown on Figure 2–1. However, on Figure 2–2, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–2 Architecture for Updating the Called Party Number by an Initial Call Control Application over a CAP Network Using Service Broker

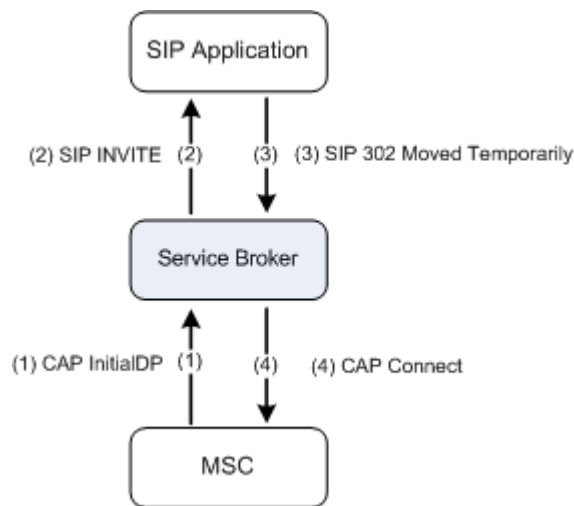
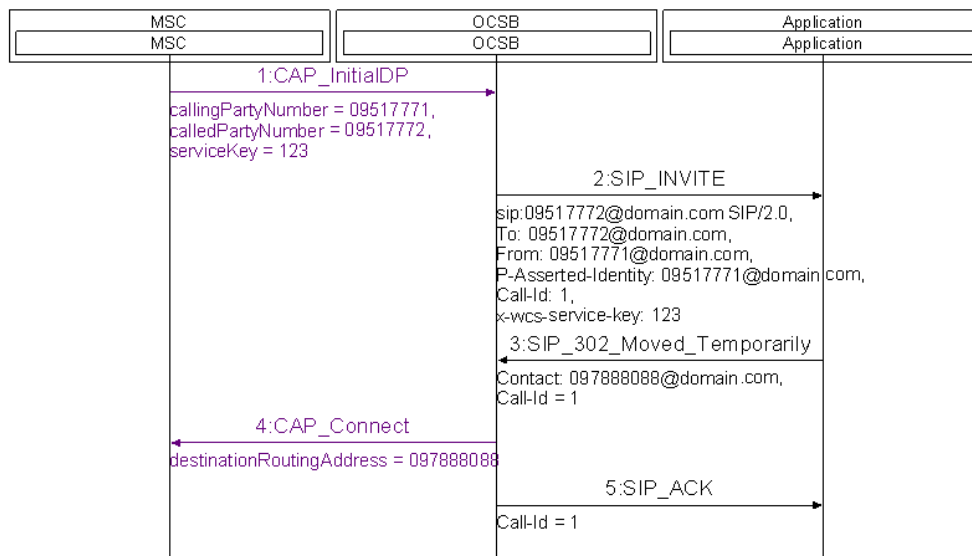


Figure 2–3 shows the detailed sequence diagram for an initial call control application that updates the called party number.

Figure 2–3 Initial Call Control Application Updates the Called Party Number



Creating CAP Connect

Service Broker creates the CAP Connect operation based on the information received in the SIP 302 Moved Temporarily response. Table 2–9 shows the content of the CAP Connect operation as set by Service Broker.

Table 2–9 Service Broker Maps SIP 302 Moved Temporarily to CAP Connect Operation

CAP Connect	Source
DestinationRoutingAddress :: AddressSignal	302 Moved Temporarily :: Contact :: user part

Table 2–9 (Cont.) Service Broker Maps SIP 302 Moved Temporarily to CAP Connect

CAP Connect	Source
DestinationRoutingAddress :: NatureOfAddress	302 Moved Temporarily :: Contact :: noa Service Broker sets the NatureOfAddress in the Connect operation based on the noa token, as set by the application in the Contact header of SIP 302 Moved Temporarily. For more information on noa values, see Table 2–4 .
DestinationRoutingAddress :: InternalNetworkNumberIndicator	Service Broker configuration
DestinationRoutingAddress :: NumberingPlanIndicator	Service Broker configuration

Leaving the Called Party Number Unmodified

Table 2–10 Leaving the Called Party Number Unmodified: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

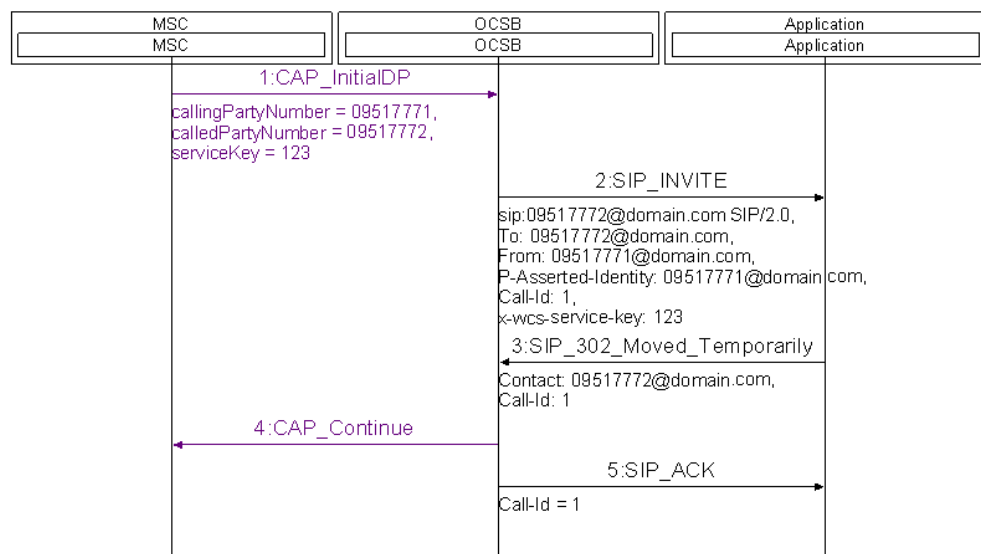
To leave the called party number unmodified, the application sets the SIP 302 Moved Temporarily response to the address provided in the user part of the Request-URI header of the SIP INVITE, which is sent by Service Broker. This address is set in the user part of the Contact header.

This action makes Service Broker to respond to a CAP InitialDP with a CAP Continue operation.

The Continue operation has no parameters.

[Figure 2–4](#) shows the detailed sequence diagram for an initial call control application that leaves the called party number unmodified.

Figure 2–4 Initial Call Control Application Leaves the Called Party Number Unmodified



Updating the Calling Party Number

Table 2–11 Updating the Calling Party Number: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To update the calling party number, the application responds with a SIP 302 Moved Temporarily message. This message contains a new calling party number encapsulated into the message body and encoded in the XER format as follows:

Example 2–1 Updating Calling Party Number in SIP 302 Moved Temporarily

```
<genericNumbers>
  <OCTET_STRING>0684136307 45716909</OCTET_STRING>
</genericNumbers>
```

After Service Broker receives SIP 302 Moved Temporarily message, Service Broker sets the genericNumbers parameter of the CAP Connect operation to the value set in genericNumbers of SIP 302 Moved Temporarily and sends the CAP Connect to MSC.

Figure 2–5 shows the high level architecture for an initial call control application that updates the calling party number over a SIP network.

Figure 2–5 Architecture for Updating the Calling Party Number by an Initial Call Control Application over a SIP Network

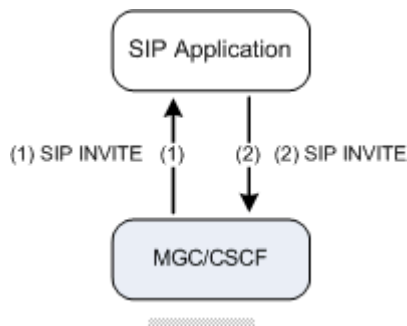


Figure 2–6 shows the same application as shown on Figure 2–5. However, on Figure 2–6, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–6 Architecture for Updating the Calling Party Number by an Initial Call Control Application over a CAP Network Using Service Broker

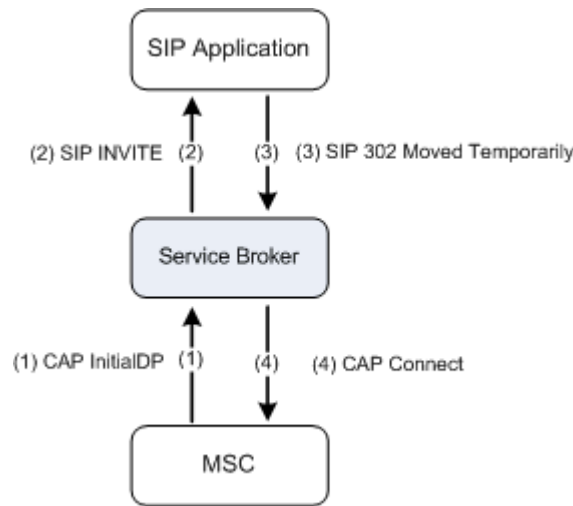
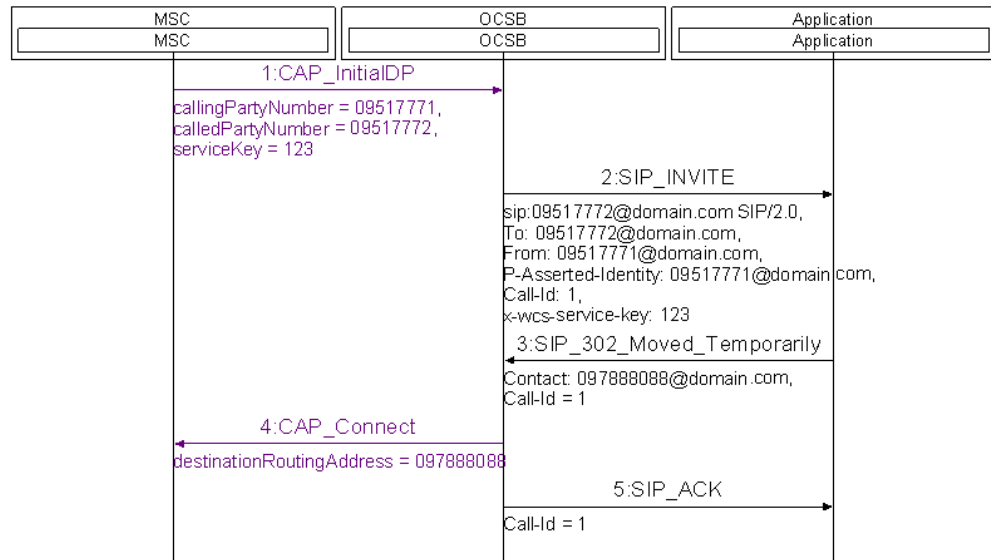


Figure 2–7 shows the detailed sequence diagram for an initial call control application that updates the calling party number.

Figure 2–7 Initial Call Control Application Updates the Calling Party Number



Developing a Full Call Control Application

Table 2–12 Developing a Full Call Control Application: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To provide a full call control, the application implements a SIP B2BUA. The application receives the SIP INVITE message sent by Service Broker and creates a new SIP dialog by sending a new SIP INVITE towards Service Broker.

Service Broker receives the SIP INVITE and sends a CAP Continue or a CAP Connect operation accompanied by a CAP RequestReportBCSEvent operation.

The CAP RequestReportBCSEvent operation instructs the MSC to monitor the call for call related events (for example O_Busy or O_No_Answer) and send notifications to Service Broker when an event is detected.

Service Broker sets the specific events to be monitored in the CAP RequestReportBCSEvent operation as defined in the Service Broker configuration.

Note: Service Broker enables the application to specify events to be monitored, and by doing this, to overwrite the Service Broker configuration. For more information, see "[Controlling the EDPs Arming](#)".

The following sections describe various call control capabilities. To provide each of the call control capabilities defined below, the application must follow relevant instructions described in the following sections.

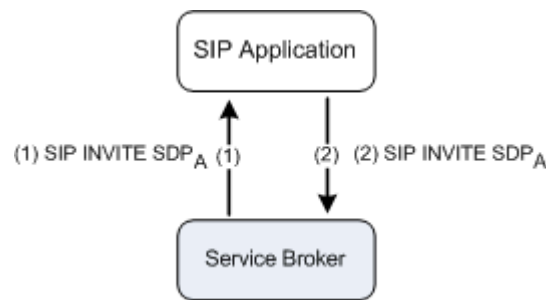
Handling the SDP

Table 2–13 Handling the SDP: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

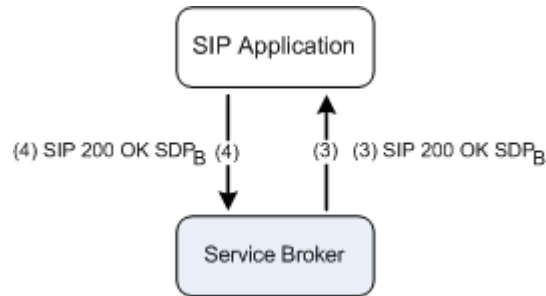
A full control application needs to propagate the SDP which is provided by Service Broker, back-to-back. [Figure 2–8](#) shows how the SDP is handled during the call initiation phase.

Figure 2–8 Architecture for Handling an SDP (Call Initiation Phase)



[Figure 2–9](#) shows how the SDP is handled during the call answering phase.

Figure 2–9 Architecture for Handling an SDP (Call Answering Phase)



Handling the SIP Route Header

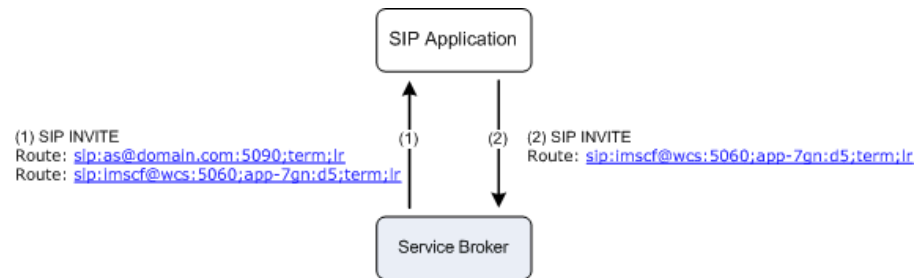
Table 2–14 Handling the SIP Route Header: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

The SIP Route header is defined in RFC 3323. A SIP full call control application implemented over the Service Broker has to follow the loose-routing mechanism defined in RFC 3261.

Figure 2–10 demonstrates the loose-routing mechanism.

Figure 2–10 Architecture for the SIP Loose Routing Mechanism



Updating the Called Party Number

Table 2–15 Updating the Called Party Number: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To update the called party number, application sets the SIP INVITE which is sent to Service Broker, to a new destination address. The new destination address is set in the user part of the RequestURI header. This makes Service Broker to respond to the CAP InitialDP with a CAP Connect operation.

Figure 2–11 shows the high level architecture for a full control application that updates the called party number.

Figure 2–11 Architecture for Updating the Called Party Number by a Full Call Control Application over a SIP Network

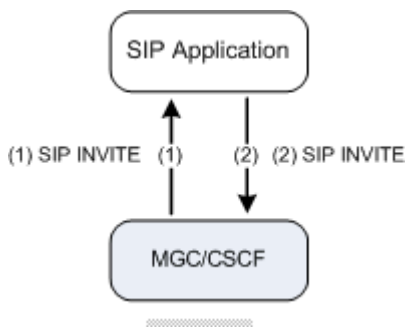


Figure 2–12 shows the same application as shown on Figure 2–11. However, on Figure 2–12, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–12 Architecture for Updating the Called Party Number by a Full Call Control Application over a CAP Network Using Service Broker

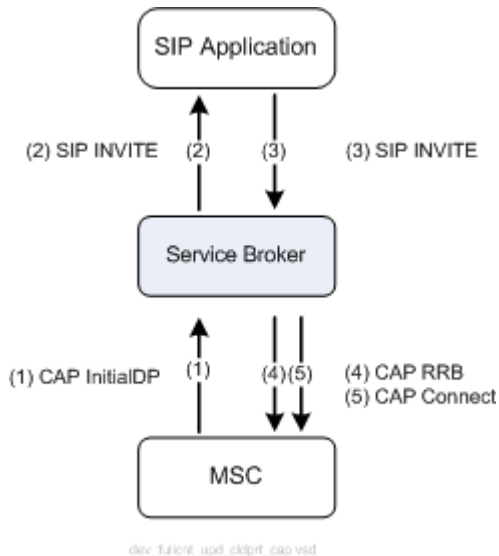


Figure 2–13 and Figure 2–14 show the detailed sequence diagram for a full control application that updates the called party number.

Figure 2–13 Full Call Control Application Updates the Called Party Number

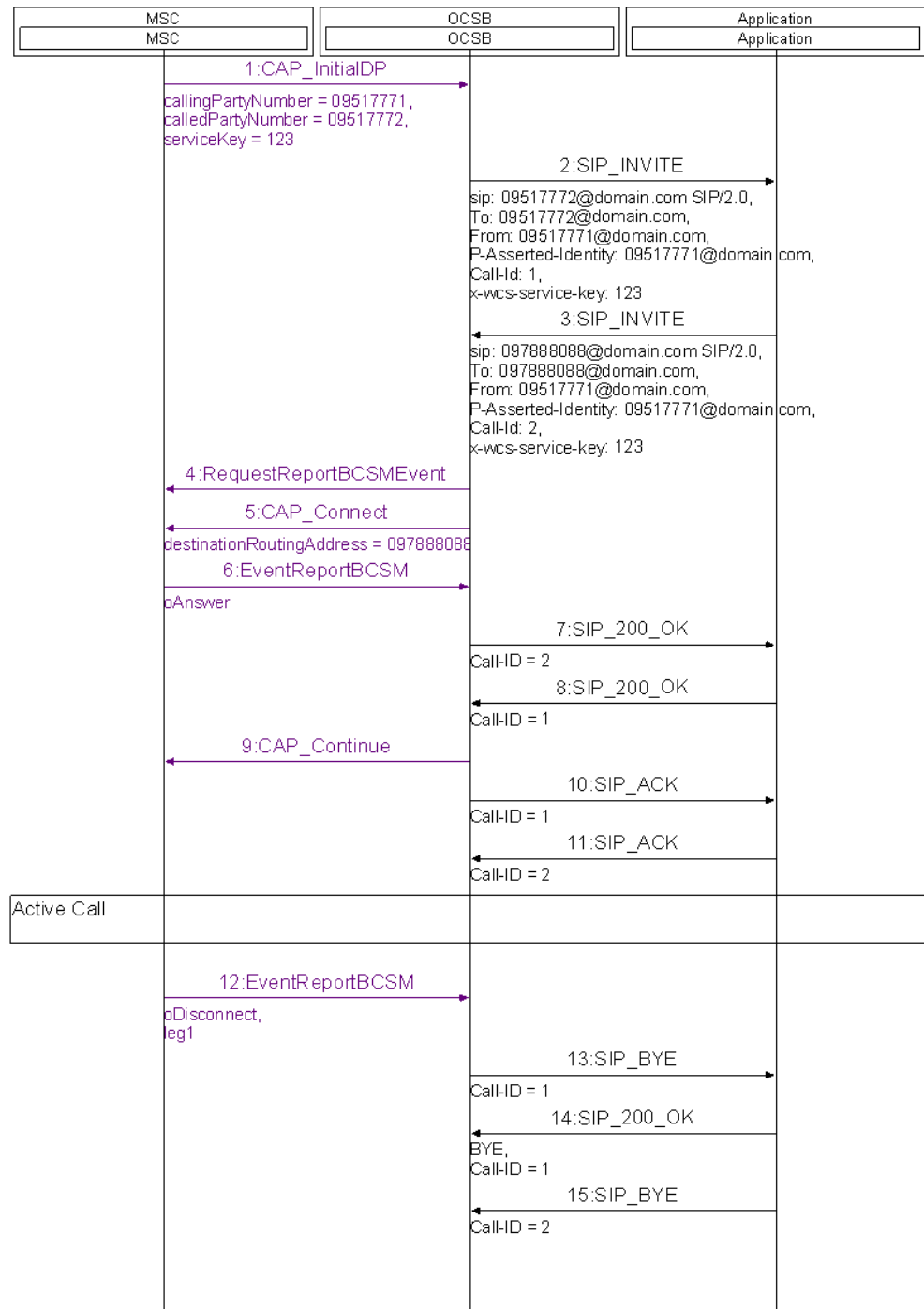
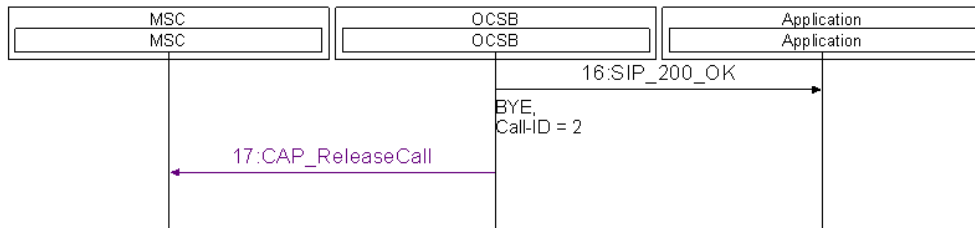


Figure 2–14 Full Call Control Application Updates the Called Party Number (cont'd)



An application can trigger Service Broker to create a CAP Connect operation using one of the following methods:

- The application can transfer the information that Service Broker uses to generate a CAP Connect in the headers of the SIP INVITE message. In this case, Service Broker maps the contents of these headers to the CAP Connect. [Table 2–16](#) shows the contents of the CAP Connect operation as set by Service Broker.

Table 2–16 Service Broker Maps SIP INVITE to CAP Connect Operation

CAP Connect	Source
DestinationRoutingAddress :: AddressSignal	INVITE :: Request-URI :: user part
DestinationRoutingAddress :: NatureOfAddress indicator	INVITE :: Request-URI :: noa token For more information on the noa token, see "Exposing Nature of Address" and "Updating the Nature of Address" .
DestinationRoutingAddress :: InternalNetwork NumberIndicator	Service Broker configuration
DestinationRoutingAddress :: NumberingPlan Indicator	Service Broker configuration
CallingPartysCategory	INVITE :: CPC
OriginalCalledPartyID	Diversion :: name-addr OriginalCalledPartyID is set from the bottom most Diversion header.
RedirectingPartyID	Diversion :: name-addr RedirectingPartyID is set from the top most Diversion header.
RedirectionInformation :: counter	Diversion :: counter The counter is set from the top most Diversion header.
RedirectionInformation :: RedirectingReason	Diversion :: reason The RedirectingReason is set from the top most Diversion header.
RedirectionInformation :: OriginalRedirectionReason	Diversion :: reason OriginalRedirection Reason is set from the bottom most Diversion header.
RedirectionInformation :: RedirectingIndicator	RedirectingIndicator is set to "call diverted".
GenericNumbers :: AddressSignal	INVITE :: From :: user part
GenericNumbers :: NumberQualifierIndicator	Service Broker configuration

Table 2–16 (Cont.) Service Broker Maps SIP INVITE to CAP Connect Operation

CAP Connect	Source
GenericNumbers :: NatureOfAddressIndicator	INVITE :: From :: noa token For more information on the noa token, see "Exposing Nature of Address" and "Updating the Nature of Address" .
GenericNumbers :: NumberIncompleteIndicator	Service Broker configuration
GenericNumbers :: NumberingPlanIndicator	Service Broker configuration
GenericNumbers :: AddressPresentationRestrictedIndicator	Service Broker configuration
GenericNumbers :: ScreeningIndicator	Service Broker configuration
ServiceInteractionIndicatorTwo :: ForwardServiceInteractionInd :: CallingPartyRestrictionIndicator	CallingPartyRestrictionIndicator is set to "presentation restricted" if INVITE :: Privacy set to "id" and INVITE :: From is set to: "Anonymous" sip:anonymous@anonymous.invalid.

- The application can encapsulate a CAP Connect operation into a SIP INVITE in the XER format. The XER must contain the destinationRoutingAddress and callingPartysCategory fields.

Leaving the Called Party Number Unmodified

Table 2–17 Leaving the Called Party Number Unmodified: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To leave the called party number unmodified, the application sets the SIP INVITE, which is sent to Service Broker, to the address provided in the SIP INVITE message received from Service Broker.

This procedure is done by copying the user part of the Request-URI of the received SIP INVITE and pasting it into the SIP INVITE sent to Service Broker. This makes Service Broker to respond to InitialDP with a CAP Continue operation.

[Figure 2–15](#) and [Figure 2–16](#) show the detailed sequence diagram for a full control application that leaves the called party number unmodified.

Figure 2–15 Full Call Control Application Leaves the Called Party Number Unmodified

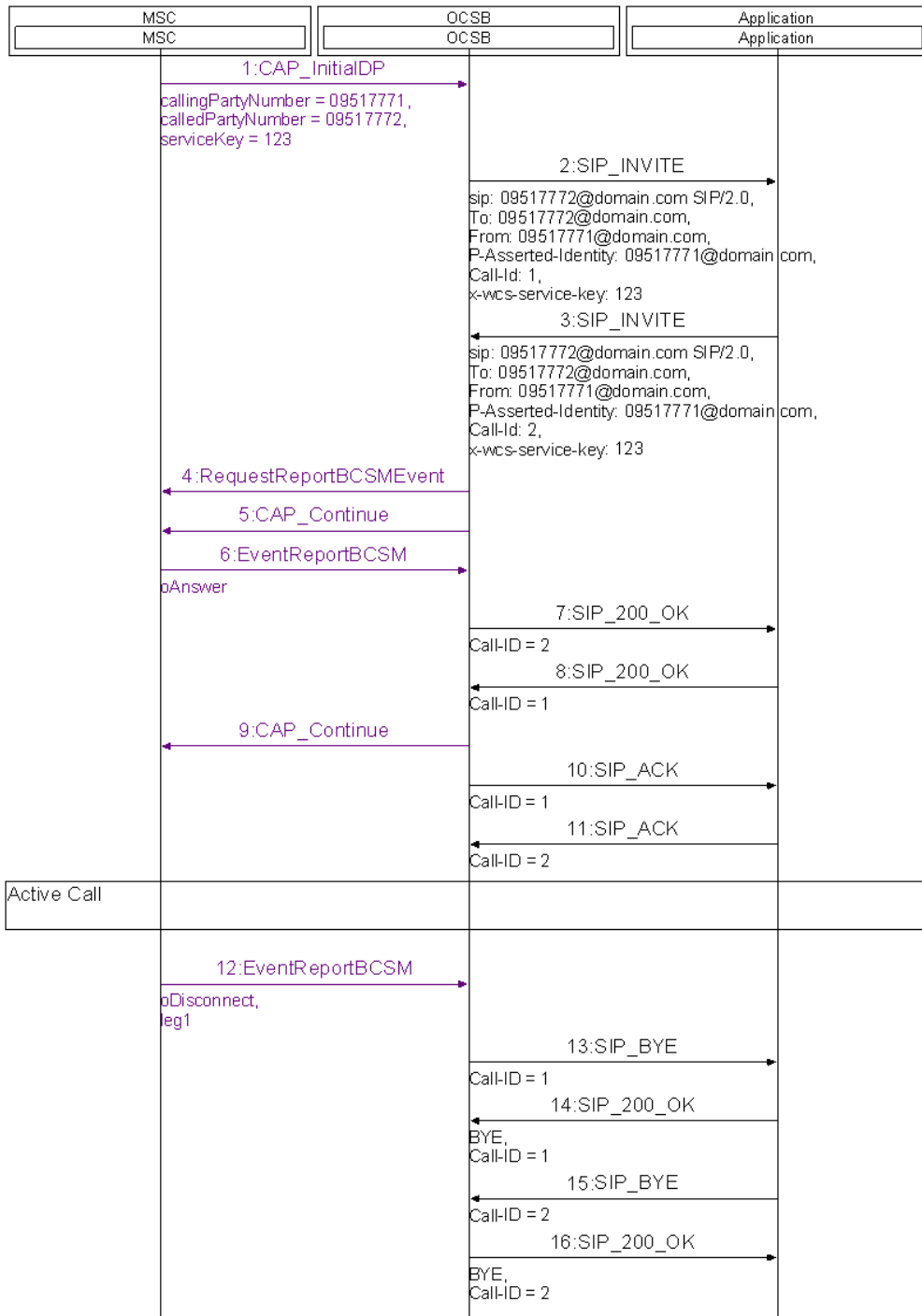
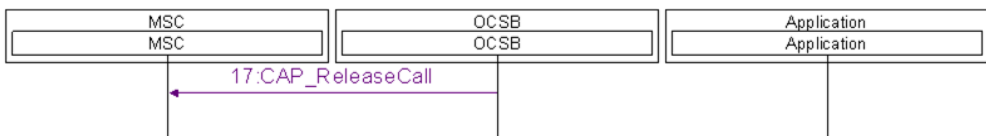


Figure 2–16 Full Call Control Application Leaves the Called Party Number Unmodified (cont'd)



Updating the Calling Party Number

Table 2–18 Updating the Calling Party Number: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To update the calling party number, the application responses with a SIP 302 Moved Temporarily message. This message contains a new calling party number encapsulated into the message body and encoded in the XER format as follows:

Example 2–2 Updating Calling Party Number in SIP 302 Moved Temporarily

```
<genericNumbers>
  <OCTET_STRING>0684136307 45716909</OCTET_STRING>
</genericNumbers>
```

After Service Broker receives SIP 302 Moved Temporarily message, Service Broker sets the genericNumbers parameter of the CAP Connect operation to the value set in genericNumbers of SIP 302 Moved Temporarily and sends the CAP Connect to MSC.

Figure 2–17 shows the high level architecture for a full call control application that updates the calling party number over a SIP network.

Figure 2–17 Architecture for Updating the Calling Party Number by a Full Call Control Application over a SIP Network

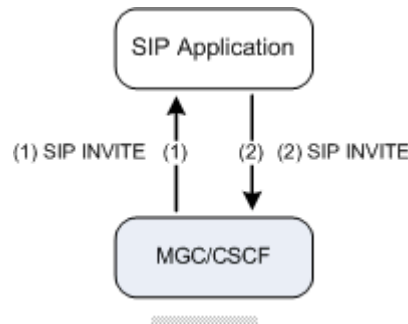


Figure 2–18 shows the same application as shown on Figure 2–17. However, on Figure 2–18, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–18 Architecture for Updating the Calling Party Number by a Full Call Control Application over a CAP Network Using Service Broker

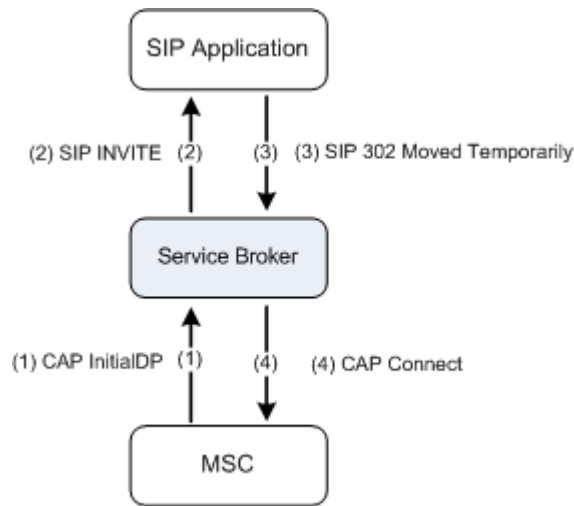
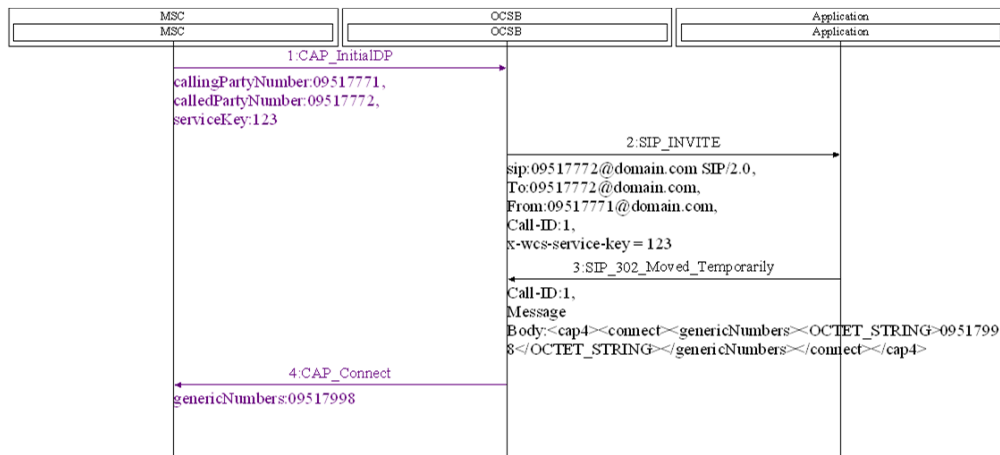


Figure 2–19 shows the detailed sequence diagram for a full call control application that updates the calling party number.

Figure 2–19 Full Call Control Application Updates the Calling Party Number



Updating the Nature of Address

Table 2–19 Updating the Nature of Address: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

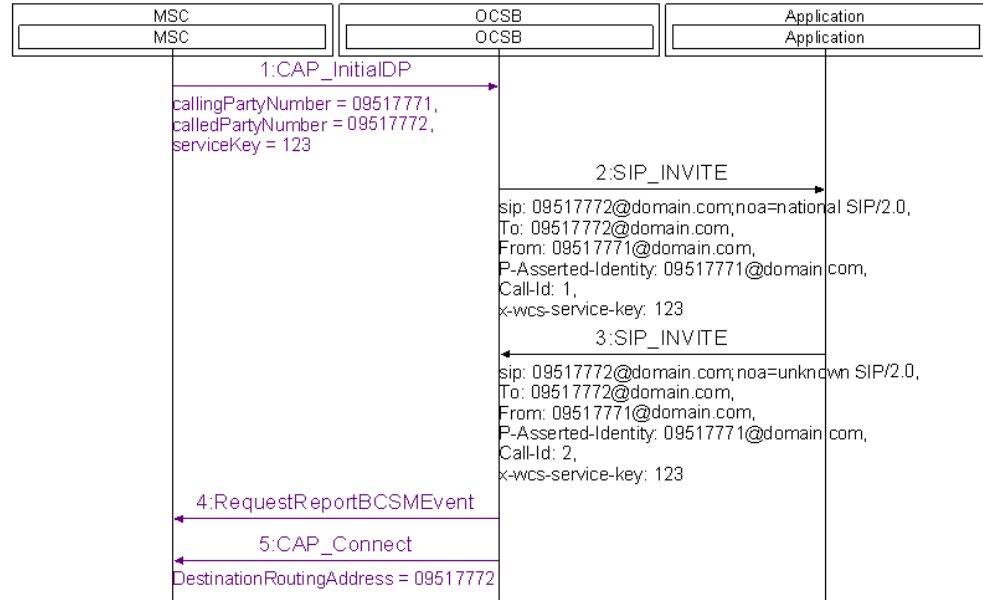
To update the nature of address of a call party, the application sets the noa token in the SIP INVITE message, which is sent to Service Broker, to the required value as follows:

- To update the called party nature of address, the application sets the noa token in the Request-URI header to the required value.

Figure 2–20 shows an example in which the application updates the called party nature of address. This example assumes that the CalledPartyNumber in CAP InitialDP is set with NatureOfAddress of type “national”. The application updates the

called party nature of address and sets it to "unknown". This causes Service Broker to set the DestinationRoutingAddress in the CAP Connect operation to NatureOfAddress of type "unknown".

Figure 2–20 Application Updates the Called Party Nature of Address



Note: In the example shown on [Figure 2–20](#), although the application does not update the called party number, Service Broker uses CAP Connect rather than CAP Continue. This is done because the Connect operation enables Service Broker to update the called party nature of address towards the MSC. CAP Connect is set to the called party number as received from the CAP InitialDP operation.

For more information on tNOA token, see ["Invoking a SIP Application"](#).

Controlling the EDPs Arming

Table 2–20 Controlling the EDPs Arming: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

As described in ["Developing a Full Call Control Application"](#), when Service Broker receives a SIP INVITE message, which is sent by a full call control application, Service Broker sends a CAP Continue or a CAP Connect operation accompanied by a CAP RequestReportBCSEvent operation.

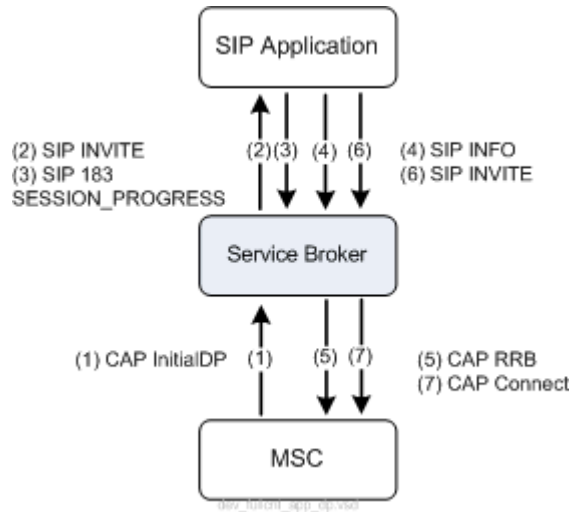
The specific events to be monitored are set in the CAP RequestReportBCSEvent operation as defined in the Service Broker configuration.

In some cases, it is required that an application dynamically controls the events that Service Broker arms for a given call, that is to define the CAP EDPs set by Service Broker in the CAP RRBCSM operation.

To control the events that Service Broker arms in the RequestReportBCSEvent operation, the application sends a SIP INFO message prior to the SIP INVITE. The SIP INFO is sent through the SIP dialog created by Service Broker and contains a XER representation of the CAP RequestReportBCSEvent operation.

Figure 2–21 shows the high level architecture for a full control application that controls the DPs armed by Service Broker.

Figure 2–21 Architecture for Controlling DPs by a Full Call Control Application



Receiving Call Events Notifications

Table 2–21 Receiving Call Events Notifications: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

Service Broker notifies a full call control application about encountered call-related events. For each call event encountered at the MSC and reported to Service Broker, Service Broker notifies the application using a corresponding SIP message as described in Table 2–22.

Notes to Table 2-22:

- RouteSelectFailure events are applicable for originating calls only.
- The table is applicable for both originating and terminating BCSM. For example, Service Broker uses SIP 486 Busy Here to notify the application about oCalledPartyBusy in an originating call and for tBusy in a terminating call.
- If a Disconnect event is reported by the calling party, Service Broker sends a SIP BYE message through the dialog created by Service Broker. If a Disconnect event is reported by the calling party, Service Broker sends a SIP BYE message through the dialog created by the application.
- For the DPs reported using SIP INFO, Service Broker sets the SIP INFO with a XER representation of the corresponding CAP EventReportBCSM operation.
- [Table 2-22](#) provides the full EDP list supported in CAP 4. Earlier CAP phases support only part of the EDPs listed in the table.

Table 2-22 Event Notifications

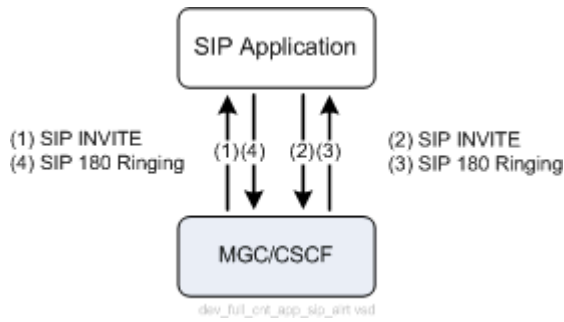
CAP event	SIP message
Route Select Failure	410 Gone
Busy	486 Busy Here
No Answer	480 Temporary Unavailable
Term Seized / Call Accepted	180 Ringing
Answer	200 OK
Disconnect	BYE
Abandon	CANCEL

To confirm notification and enable Service Broker to instruct the MSC to continue call processing at event notification, the application propagates the received SIP message back-to-back.

Note:: Call processing is suspended by the MSC when an event armed as EDP-R is encountered. When EDP-R is reported to Service Broker, MSC requests Service Broker instructions for call processing.

[Figure 2-22](#) shows a full control application in the call initiation process. When the called party is alerted, the application receives a SIP 180 Ringing message and propagate it back-to-back.

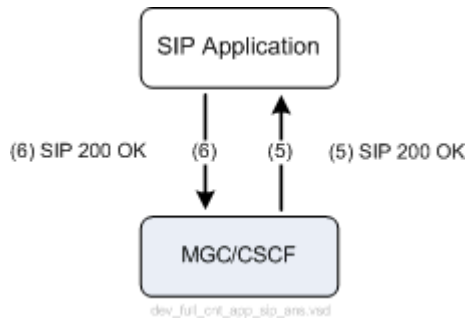
Figure 2–22 Architecture for Initiating a Call over a SIP Network (Alerting Phase)



When the called party answers the call, the application receives a SIP 200 OK and again propagates this message back-to-back towards the initiating side.

Figure 2–23 shows a full control application in the call answering phase.

Figure 2–23 Architecture for Initiating a Call over a SIP Network (Answering Phase)



Finally, when the called party (or in another scenario, the calling party) disconnects the call, the application receives a SIP BYE and propagates it towards the initiating side as shown on Figure 2–24.

Figure 2–24 Architecture for Initiating a Call over a SIP Network (Disconnecting Phase)

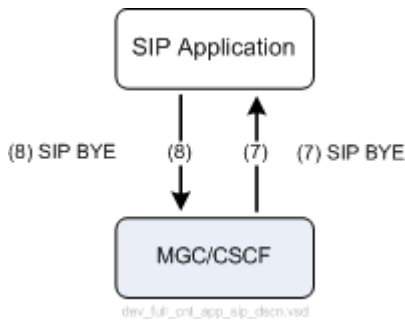


Figure 2–25, Figure 2–26, and Figure 2–27 show the same application as shown on Figure 2–22 and Figure 2–24. However, the application below provides the same functionality over a CAP network using Service Broker.

Figure 2–25 Architecture for Initiating a Call by a Full Control Application over a CAP Network Using Service Broker (Alerting Phase)

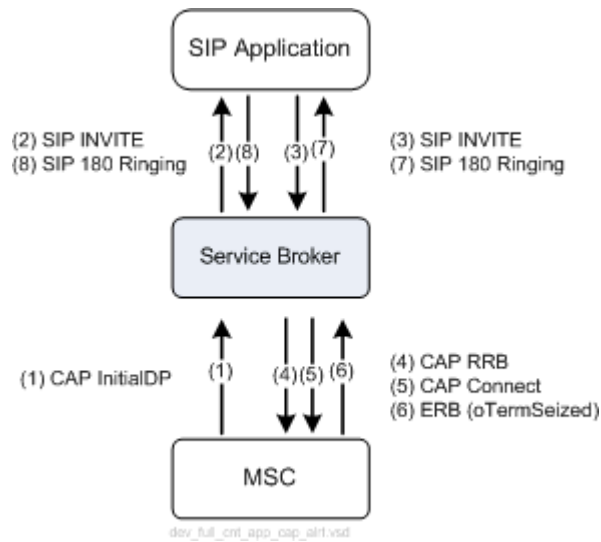


Figure 2–26 Architecture for Initiating a Call by a Full Control Application over a CAP Network Using Service Broker (Answering Phase)

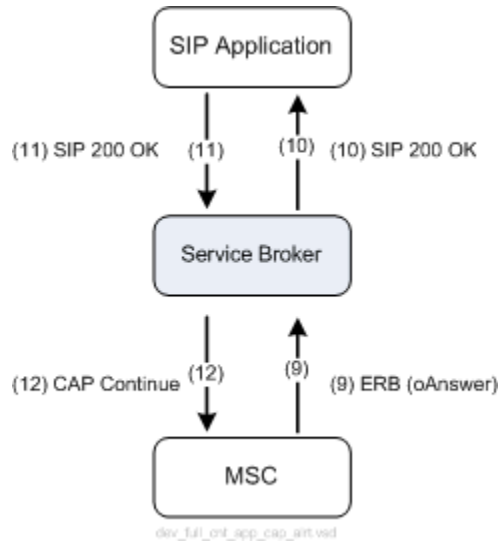


Figure 2–27 Architecture for Initiating a Call by a Full Control Application over a CAP Network Using Service Broker (Disconnecting Phase)

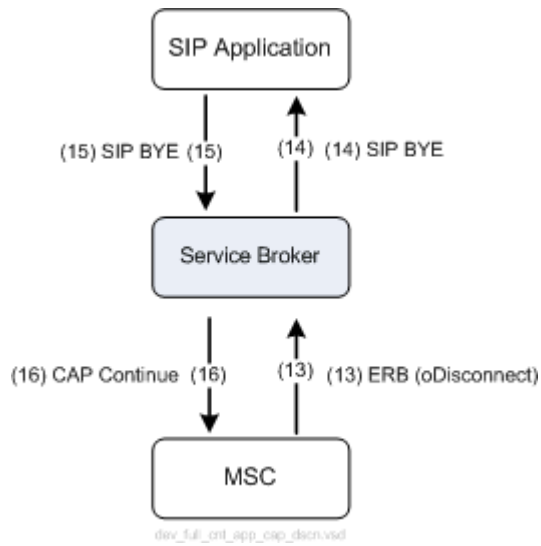
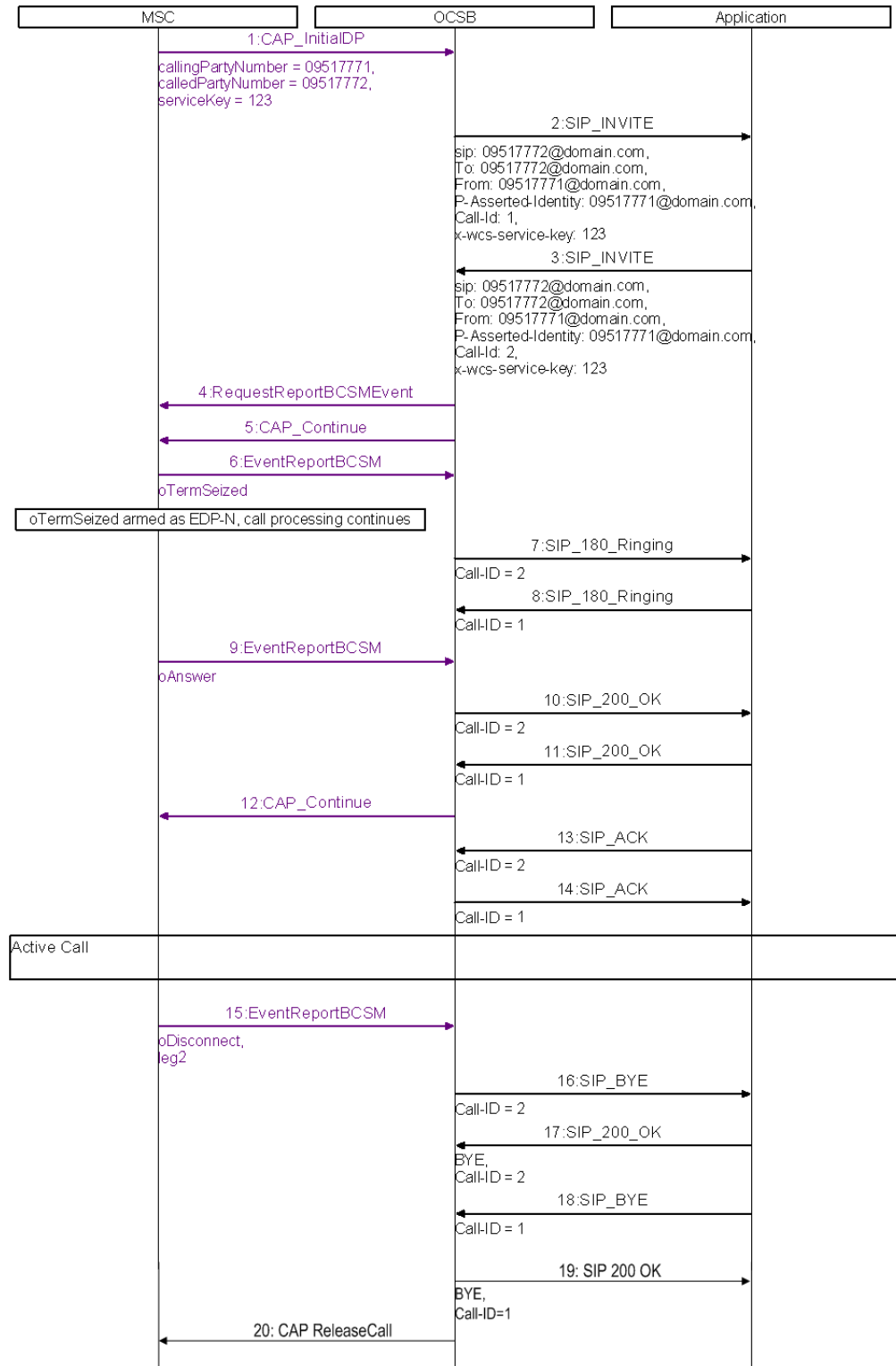


Figure 2–28 shows the detailed sequence diagram that demonstrates how Service Broker notifies the application for alerting, answer, and disconnect events.

Figure 2–28 Service Broker Notifies Application: Alerting, Answering, and Disconnecting Phases



Terminating a Call

Table 2–23 Terminating a Call: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To terminate a call, the application sends a SIP BYE request towards Service Broker. The BYE request is sent on both active dialogs, that is the dialog created by Service Broker and the dialog created by the application.

Service Broker uses the BYE request to terminate the CAP dialog towards MSC using a CAP ReleaseCall operation.

Figure 2–29 shows architecture for a full control application terminating a call.

Figure 2–29 Architecture for Terminating a Call over a SIP Network

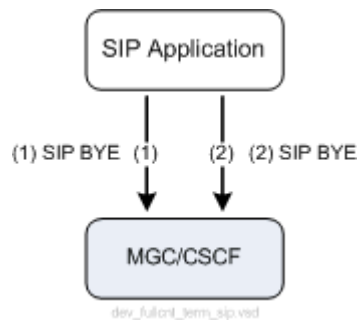
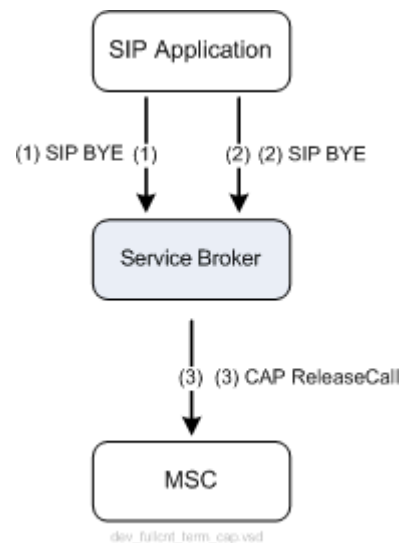


Figure 2–30 shows the same application as shown on Figure 2–29. However, on Figure 2–30, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–30 Architecture for Terminating a Call over a CAP Network Using Service Broker



Service Broker Error Responses

Service Broker may respond towards the application with a SIP error in case an application request cannot be fulfilled or in other error cases as defined in [Table 2–24](#).

Table 2–24 SIP Errors

SIP Error	Description
405 Method Not Allowed	Sent by Service Broker in case the application requests a call control operation which is not legal in the current moment on the CAP interface
403 Forbidden	Sent by Service Broker in case the application requests a call control operation which is not supported by the specific CAP interface
415 Unsupported Media Type	Sent by Service Broker in case the application provides a non-supported SDP

Rejecting a Call

Table 2–25 Rejecting a Call: Applicable CAP Phases

CAP 1	CAP 2	CAP 3	CAP 4
YES	YES	YES	YES

To reject a call, the application responds to the SIP INVITE with a SIP error response (for example, SIP 404 Not Found). When Service Broker receives the SIP error response, Service Broker performs one of the following actions:

- Instructs the MSC to terminate the call by sending a CAP ReleaseCall operation
- Instructs the MSC to allow the call to continue by sending a CAP Continue operation

Service Broker determines the action to be performed based on its configuration.

[Figure 2–31](#) shows the high level architecture for a full control application that terminates a call.

Note: The figures below shows an example in which the application uses SIP 404 Not Found to reject the call. In practice, applications are not limited to a specific SIP error response.

Figure 2–31 Architecture for Rejecting a Call over a SIP Network

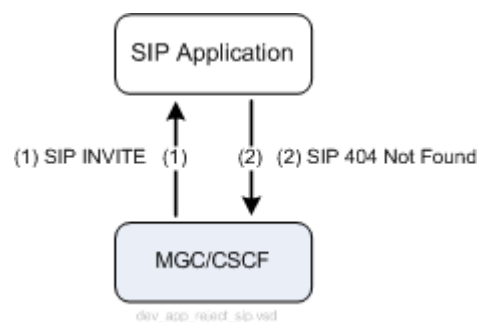
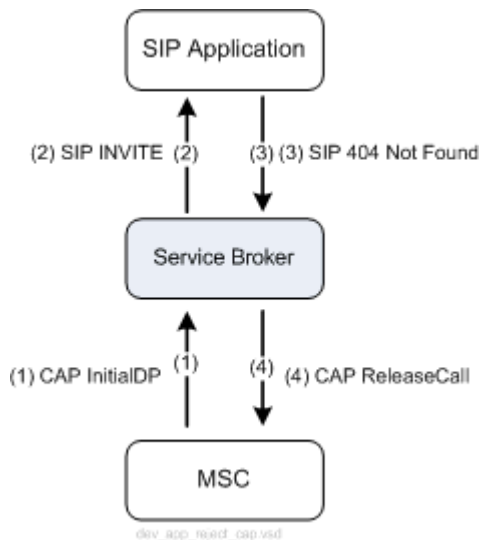


Figure 2–32 shows the same application as shown on Figure 2–31. However, on Figure 2–32, the application provides the same functionality over a CAP network using Service Broker.

Figure 2–32 Architecture for Rejecting a Call over a CAP Network Using Service Broker



Controlling the CAP Release Cause

A SIP application can control how Service Broker generates the **cause** parameter of a CAP Release message using the following methods:

- Setting the **Reason** header of a SIP BYE or SIP CANCEL message. This header should contain a specific cause that IM-SCF can use. For example: Reason: Q.850; cause:31; text: "Session terminated". See RFC 3326, The Reason Header Field for the Session Initiation Protocol (SIP), for more information about the Reason header.
- Sending a SIP error response to Service Broker. Service Broker uses this response to set the cause parameter in the CAP ReleaseCall operation. To instruct Service Broker to set the cause parameter of a ReleaseCall operation to a specific value, the application uses the corresponding SIP error response as defined in Table 2–26.

Table 2–26 Release Cause

CAP Release Cause	SIP Response
31, normal unspecified	400 to 479
19, no answer from user	480
31, normal unspecified	481 to 485
17, user busy	486
31, normal unspecified	487 to 699
31, normal unspecified	Any other SIP error response

Developing a Charging Application

This chapter describes how to develop a SIP application that performs CAP charging operations.

About CAP Charging Operations

Service Broker enables a SIP application to perform CAP charging operations by attaching a XER formatted body to a SIP message. The XER body is an XML representation of the CAP charging operation to be performed.

For example, an application may instruct Service Broker to send a CAP FurnishChargingInformation towards an MSC by sending a SIP INFO message towards Service Broker. The SIP INFO message includes an XML representation of CAP FurnishChargingInformation carried by the message body.

Note: Some of the charging operations, such as ApplyCharging, can be used only by a full call control application, while others, such as FurnishChargingInformation, can be used by an initial call control application as well.

Performing FurnishChargingInformation Operation

To perform a CAP FurnishChargingInformation operation, the application sends a SIP INFO message through the SIP dialog created by Service Broker. The application attaches a XER representation of FurnishChargingInformation operation to the SIP INFO body.

[Figure 3-1](#) shows a SIP initial call control application that performs a CAP FurnishChargingInformation operation.

Figure 3–1 Initial Call Control Application Performs CAP FurnishChargingInformation

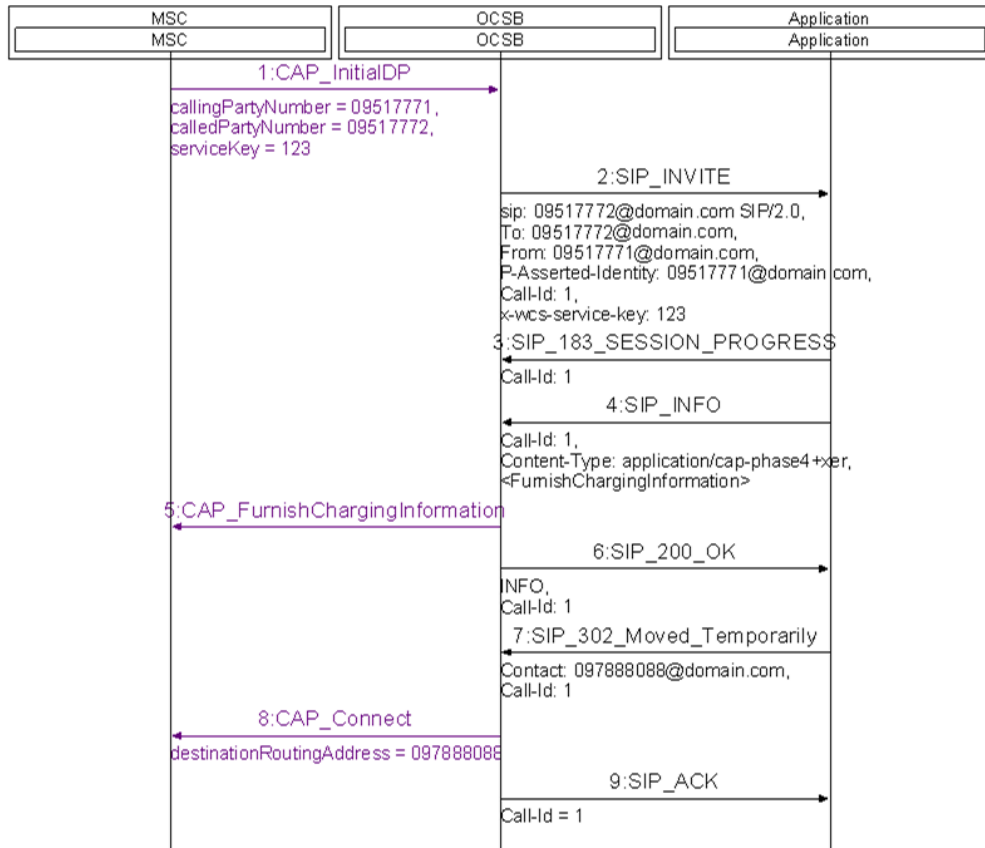


Figure 3–2 and Figure 3–3 show a SIP full call control application that performs a CAP FurnishChargingInformation operation.

Figure 3–2 Full Call Control Application Performs CAP FurnishChargingInformation

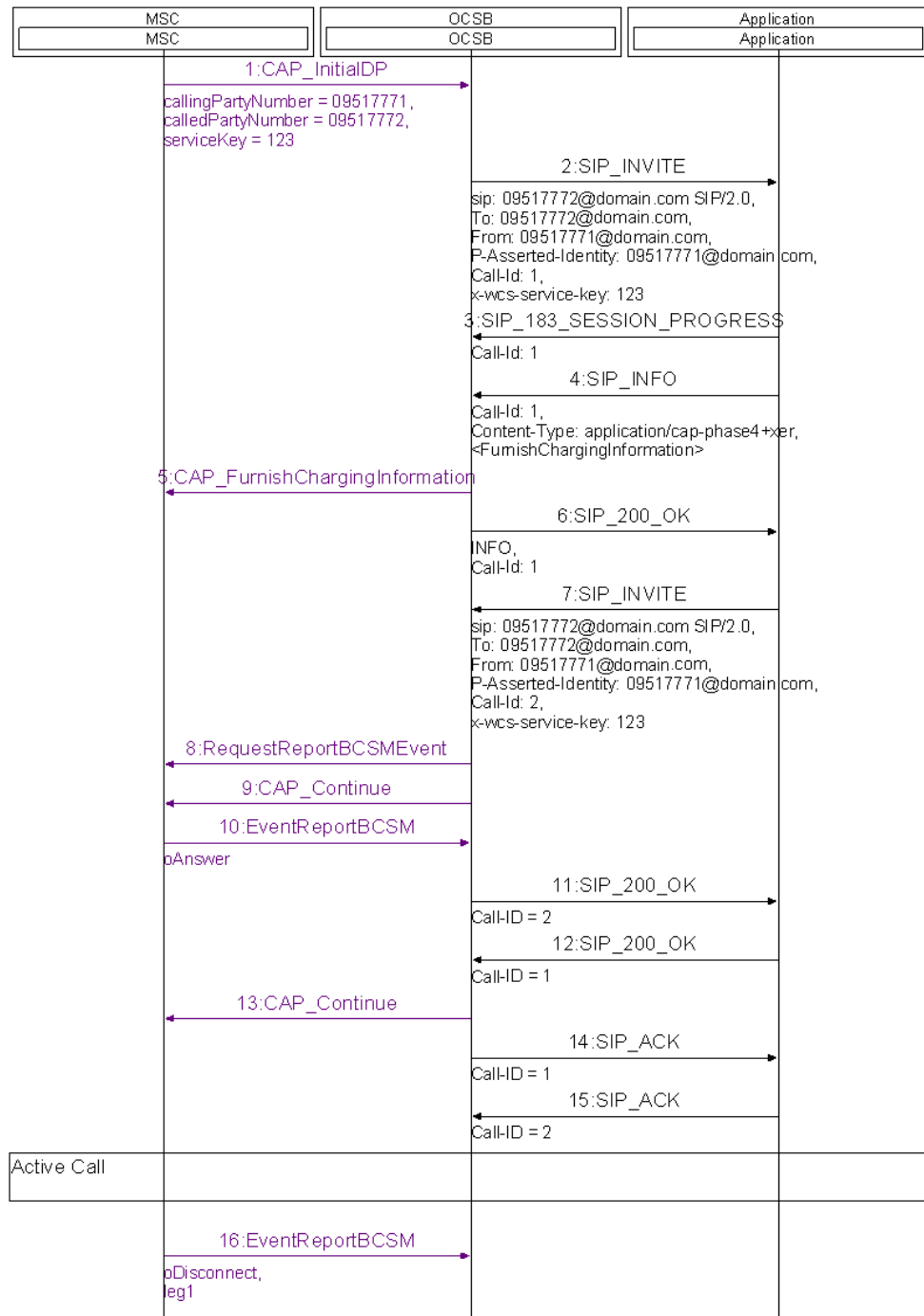
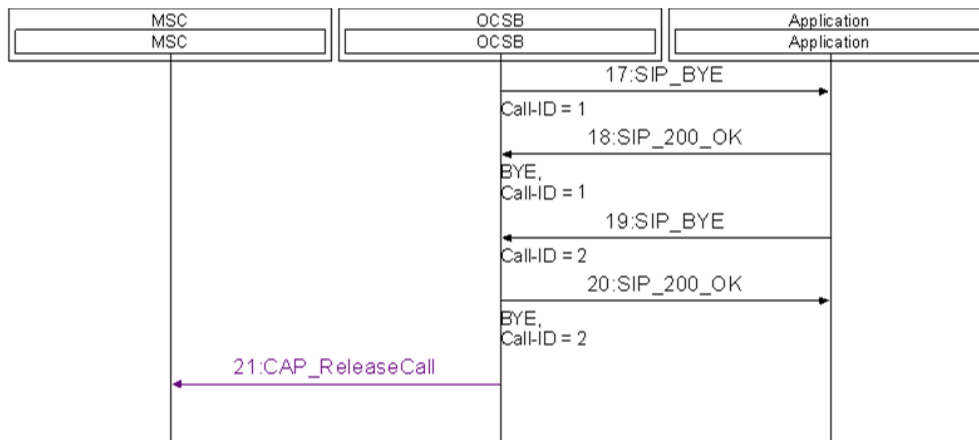


Figure 3–3 Full Call Control Application Performs CAP FurnishChargingInformation (cont'd)



Performing FurnishChargingInformation in an Application Initiated Call

To perform a CAP FurnishChargingInformation operation, the application sends a SIP INFO message through the SIP dialog created by the application. The application attaches a XER representation of the FurnishChargingInformation operation to the SIP INFO body.

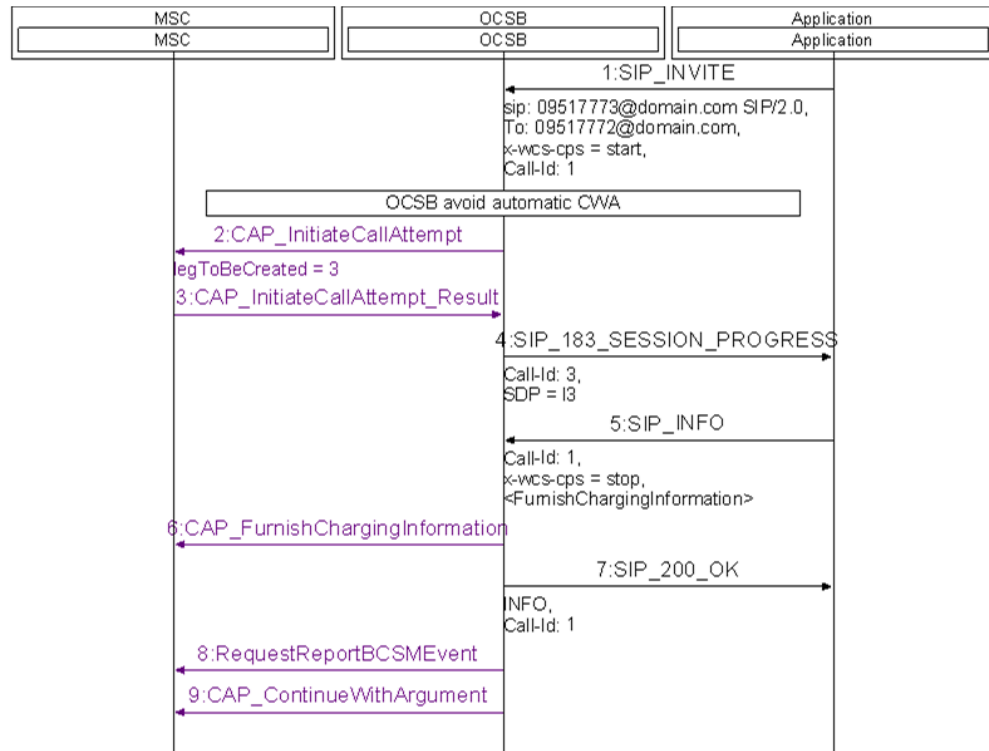
When performing CAP FurnishChargingInformation in an application initiated call, the application should perform the following steps:

1. To set INVITE sent to Service Broker (for creating the new leg) with an x-wcs-cps header that holds the value of "start"
2. To receive a SIP 183 SESSION PROGRESS message from Service Broker
3. To send a SIP INFO message (carrying the CAP FurnishChargingInformation) and set the SIP INFO with an x-wcs-cps header that holds the value of "stop"

The application uses the x-wcs-cps header (with the value of "start") to instruct Service Broker to avoid continuing call processing towards the MSC immediately after the CAP InitiateCallAttempt is sent and instead, to consider the SIP INFO sending CAP FurnishChargingInformation. The CAP ContinueWithArgument is sent by Service Broker only upon receiving an x-wcs-cps header that holds the value of "stop".

Figure 3–4 shows an application that performs a CAP FurnishChargingInformation operation in an application initiated call.

Figure 3–4 Application Performs CAP FurnishChargingInformation in an Application Initiated Call



Performing SendChargingInformation Operation

To perform a CAP SendChargingInformation operation, the application sends a SIP INFO message through the SIP dialog created by Service Broker. The application attaches a XER representation of the CAP SendChargingInformation operation to the SIP INFO body.

Performing ApplyCharging Operation

To perform a CAP ApplyCharging operation, the application sends a SIP INFO message through the SIP dialog created by Service Broker. The application attaches a XER representation of ApplyCharging operation to the SIP INFO body.

The following example shows a XER representation of the CAP phase 4 ApplyCharging operation.

```

<Cap4>
  <applyCharging>
    <aChBillingChargingCharacteristics>
      A004800204B0
    </aChBillingChargingCharacteristics>
  </applyCharging>
</Cap4>
  
```

Receiving a Report

A SIP application that performs an ApplyCharging operation expects to receive a report when, for example, the allocated call duration expires.

Service Broker sends the report by sending a SIP INFO message. Service Broker attaches a XER representation of an ApplyChargingReport operation to the SIP INFO message and sends this message through the dialog created by Service Broker.

The following example shows a XER representation of the CAP phase 4 ApplyChargingReport operation.

```
<Cap4>
  <applyChargingReport>
    A014A003 810101A1 0380012F 820100A5 05A20381 0101
  </applyChargingReport>
</Cap4>
```

The CAP phase 4 ApplyChargingReport operation includes a parameter called `legActive` that indicates whether or not the call is active at the time when the report is sent. If at the time when Service Broker is sending the report the call is still active, the application may allocate additional quota that allows the call to continue. In this case, several ApplyCharging and ApplyChargingReport operations are exchanged between the application and Service Broker.

[Figure 3–5](#) and [Figure 3–6](#) show a SIP full call control application that performs a CAP phase 4 ApplyCharging operation.

Note: In the example shown on [Figure 3–5](#), the application reserves quota for the call. When the first quota is exhausted, Service Broker sends a report towards the application which reserve an additional quota.

Figure 3-5 Full Call Control Application Performs CAP ApplyCharging

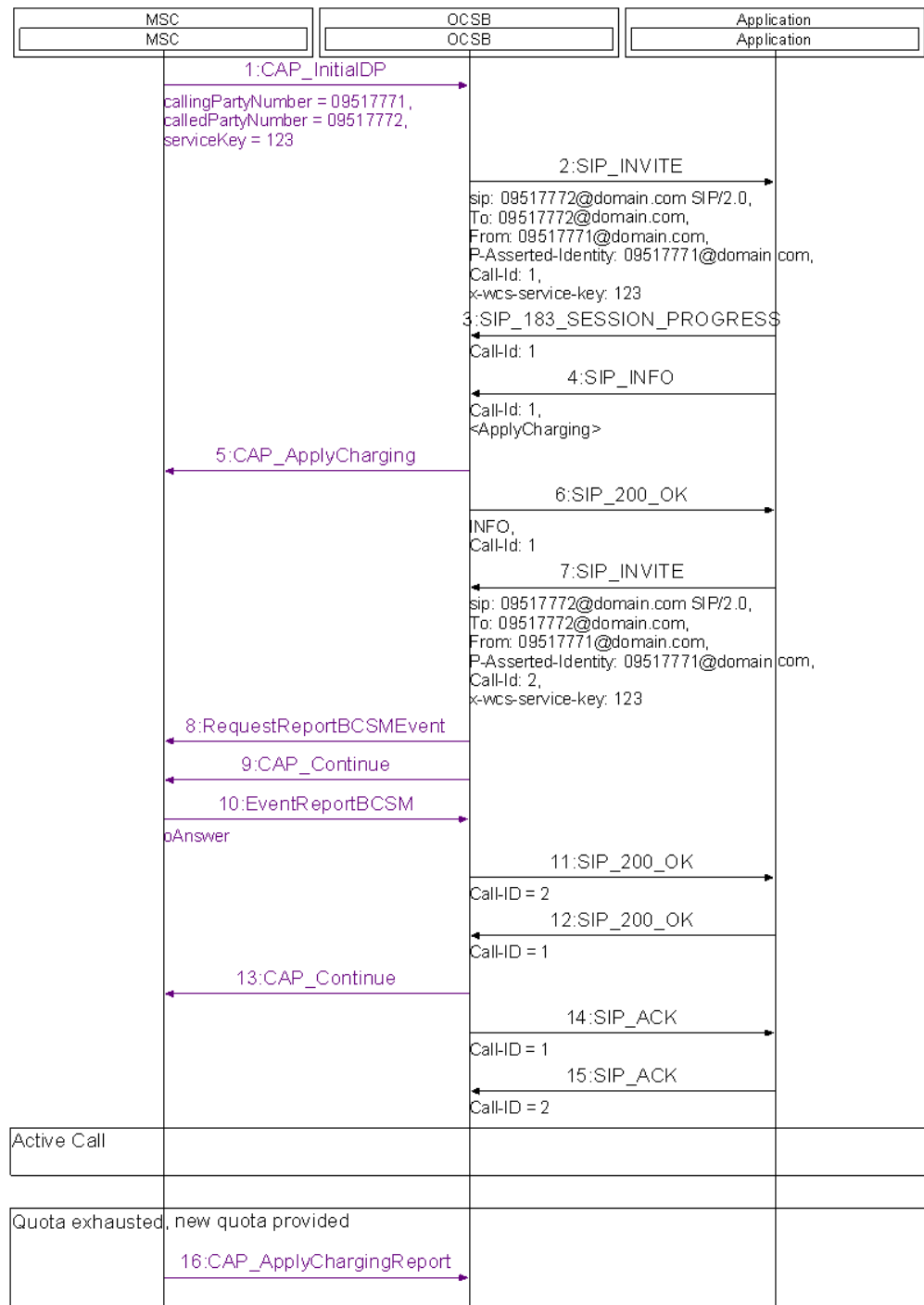
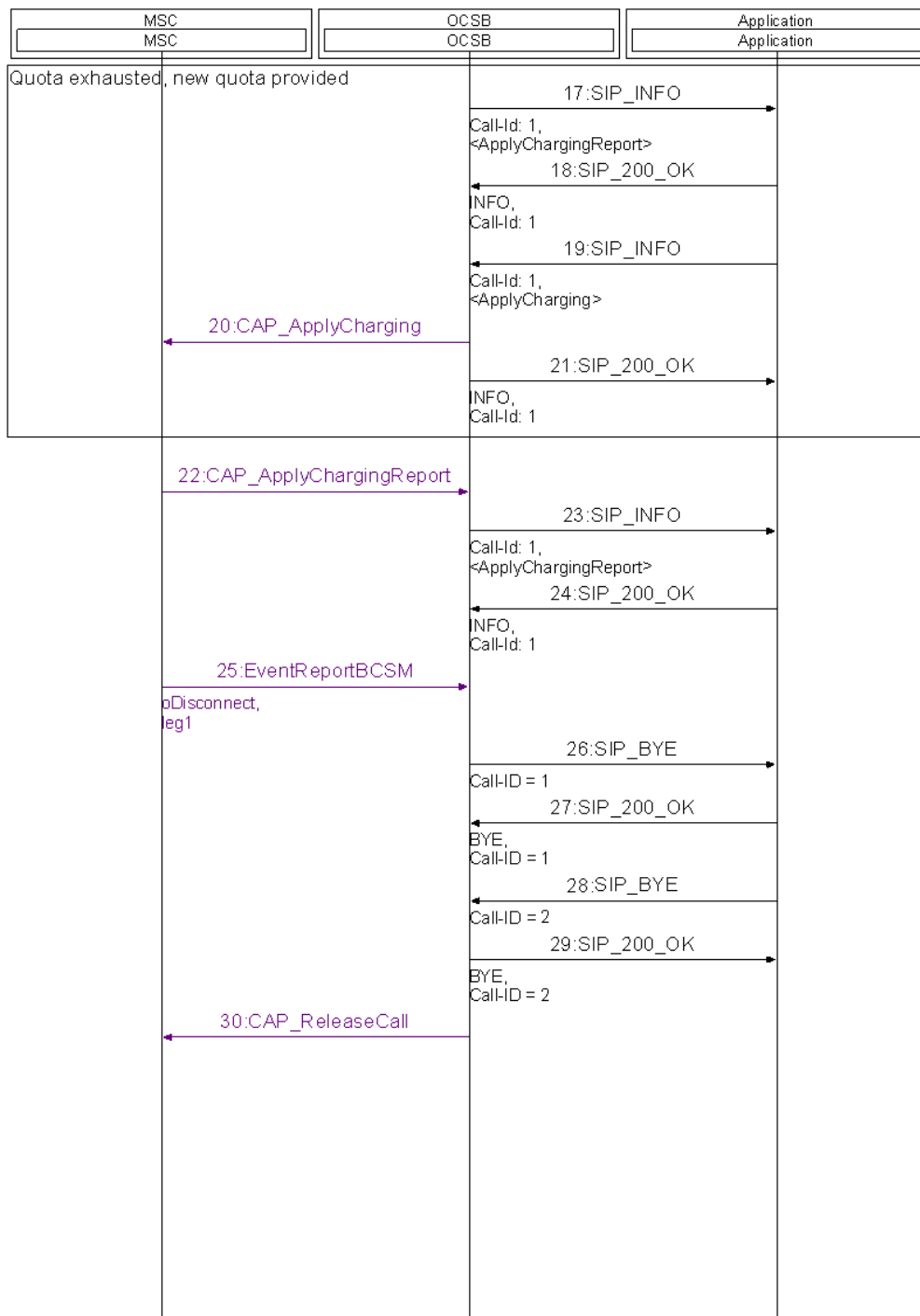


Figure 3–6 Full Call Control Application Performs CAP ApplyCharging (cont'd)



Performing CallInformationRequest Operation

To perform a CAP CallInformationRequest operation, the application sends a SIP INFO message through the SIP dialog created by Service Broker. The application attaches a XER representation of the CAP CallInformationRequest operation to the SIP INFO body.

The following example shows an XML representation of the CAP phase 4 CallInformationRequest operation.

```
<Cap4>
  <callInformationRequest>
    <requestedInformationTypeList>
      <callAttemptElapsedTime/>
    </requestedInformationTypeList>
  </callInformationRequest>
</Cap4>
```

Receiving a Report

A SIP application that performs a CallInformationRequest operation expects to receive a report holds the charging related information. Service Broker sends a report by sending a SIP INFO message. Service Broker attaches a XER representation of the CallInformationReport operation to the SIP INFO message and sends this message through the dialog created by Service Broker.

Developing a SIP User Interaction Application

This chapter describes how to develop a SIP application that provides user interaction functionality.

Providing User Interaction Functionality

User interaction is the method of having a SIP application interacting with a call party during the application logic processing.

The following two forms of user interaction exist:

- Playing an announcement, when an announcement is played to a call party
- Information collection, when an announcement is played to a call party and DTMF digits are collected from the call party

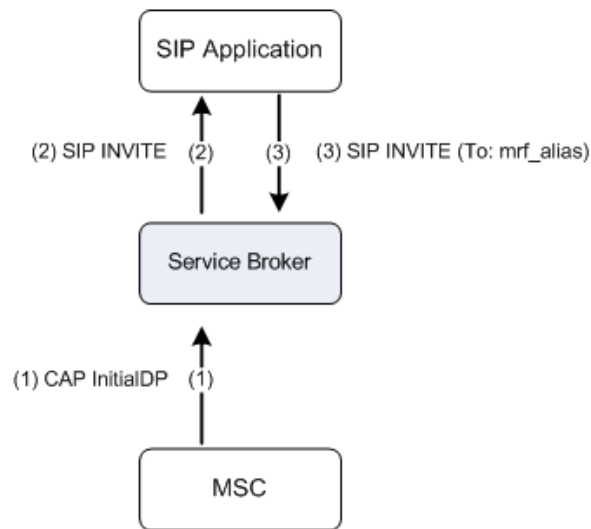
An application may perform a user interaction operation during different points of call:

- During call establishment, when the application provides an announcement to a call party prior to completing the call establishment process. This announcement is called pre-call announcement.
- During active call, when the application provides an announcement to a call party during an active call. This announcement is called mid-call announcement.
- During call release, when the application provides an announcement to a call party prior to completing the call release process. This announcement is called post-call announcement.

To provide the user interaction functionality, the application receives a SIP INVITE message sent by Service Broker, and sends a new SIP INVITE targeted to an MRF. The application performs this action by setting the To header of the SIP INVITE to the address of the MRF.

Figure 4-1 shows the high level architecture for an application that creates a user interaction session.

Figure 4–1 Architecture for Creating a User Interaction Session by the Application



About the Media Resource Table

Service Broker has a media resource table configuration. When receiving the INVITE from an application, Service Broker extracts the content of the To header and uses this content as an MRF alias. Service Broker queries the media resource table for this alias to determine the following parameters:

- Media resource type: Service Broker determines whether an external or an internal media resource is used

Note: The terms external and internal referred to MSC. An external resource is a stand-alone gsmSRF (IP) while internal resource is an gsmSRF integrated within the MSC.

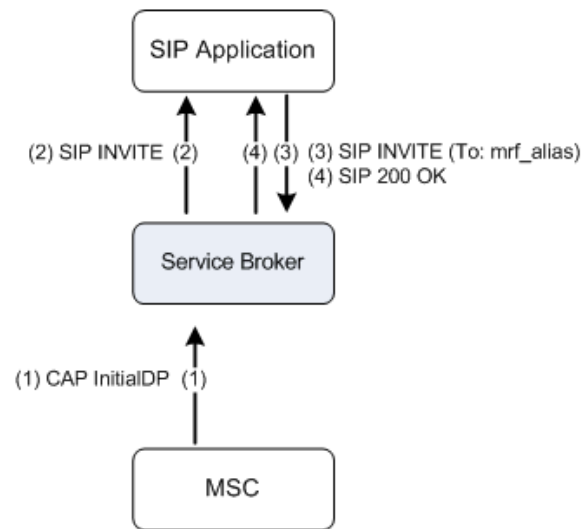
When an external resource is used, Service Broker uses a CAP EstablishTemporaryConnection (ETC) operation. When an internal resource is used, Service Broker uses a CAP ConnectToResource (CTR) operation.

- Media Resource Address: the address of the media resource
- Answer indication: a flag used by Service Broker to instruct an MSC about sending a backward answer message through the ISUP connection

Upon determination of a media resource to be used, Service Broker acts towards the SIP application as an MRF, responds to the application with SIP 200 OK. The SIP 200 OK message includes an SDP created by Service Broker.

Figure 4–2 shows the high level architecture for Service Broker that confirms the MRF dialog created by the application.

Figure 4–2 Architecture for Sending SIP 200 OK from Service Broker to the Application



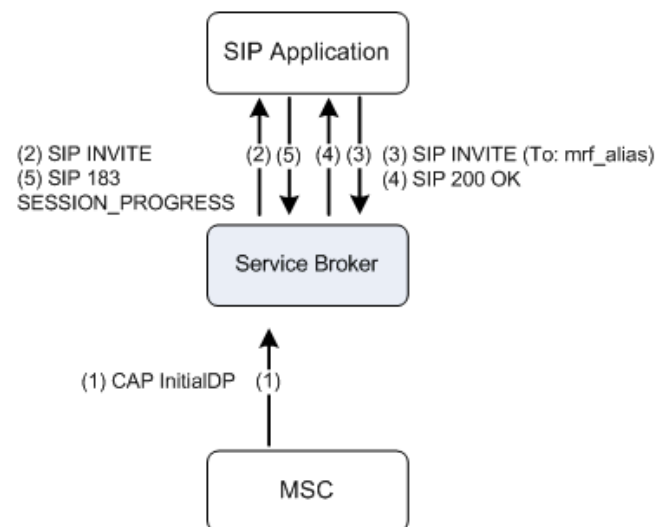
When the application receives the SIP 200 OK message, the application updates the SDP towards the initiating side. This is done by propagating the SDP received from Service Broker towards the SIP dialog created by Service Broker.

To update the SDP towards the initiating side, the application uses one of the following SIP messages:

- SIP 183 SESSION PROGRESS: usually, in a pre-call announcement scenario
- SIP reINVITE: usually, in a mid-call and post-call announcement scenario

Figure 4–3 shows the high level architecture for an application that updates the SDP towards the call initiating side (pre-call announcement scenario).

Figure 4–3 Architecture for Updating the SDP towards the Initiating Side



Based on the specific MRF alias, Service Broker performs one of the following CAP operations:

- CAP EstablishTemporaryConnection (ETC)

- CAP ConnectToResource (CTR)

Suspending Call Processing for Pre-Call Announcements

An application can create a call leg and connect this leg to a media resource for playing announcement or collecting information. To be able to play an announcement or collect information on the leg answer, the application needs to instruct Service Broker to suspend sending a ContinueWithArguments operation when Service Broker received ERB-R Answer from the leg.

While Service Broker is in the suspension mode, the application sends to Service Broker a SIP INVITE with the To header set to the alias of the MRF. After the MRF played the announcement or the leg terminated the call (see "[Terminating a User Interaction Session](#)" for more information), Service Broker sends a ContinueWithArguments to the leg.

To set Service Broker in the suspension mode on the leg answer, the application needs to set the x-wsc-cps header to "late".

[Figure 4-4](#) and [Figure 4-5](#) shows a call flow when an application creates a leg by sending a SIP INVITE message to Service Broker. To force Service Broker to suspend sending a ContinueWithArguments operation, the application sets the x-wsc-cps header to "late". After the announcement is played, and Service Broker sent the DisconnectFromResource operation, Service Broker continues the call by sending the ContinueWithArguments. Similarly, Service Broker can continue the call if the leg sends an ERB Disconnect operation.

Notice that the x-wsc-cps header set to "late" is added to the leg SIP INVITE message rather than to the SIP INVITE to the MRF alias.

Figure 4-4 Instructing Service Broker to Suspend Call Processing

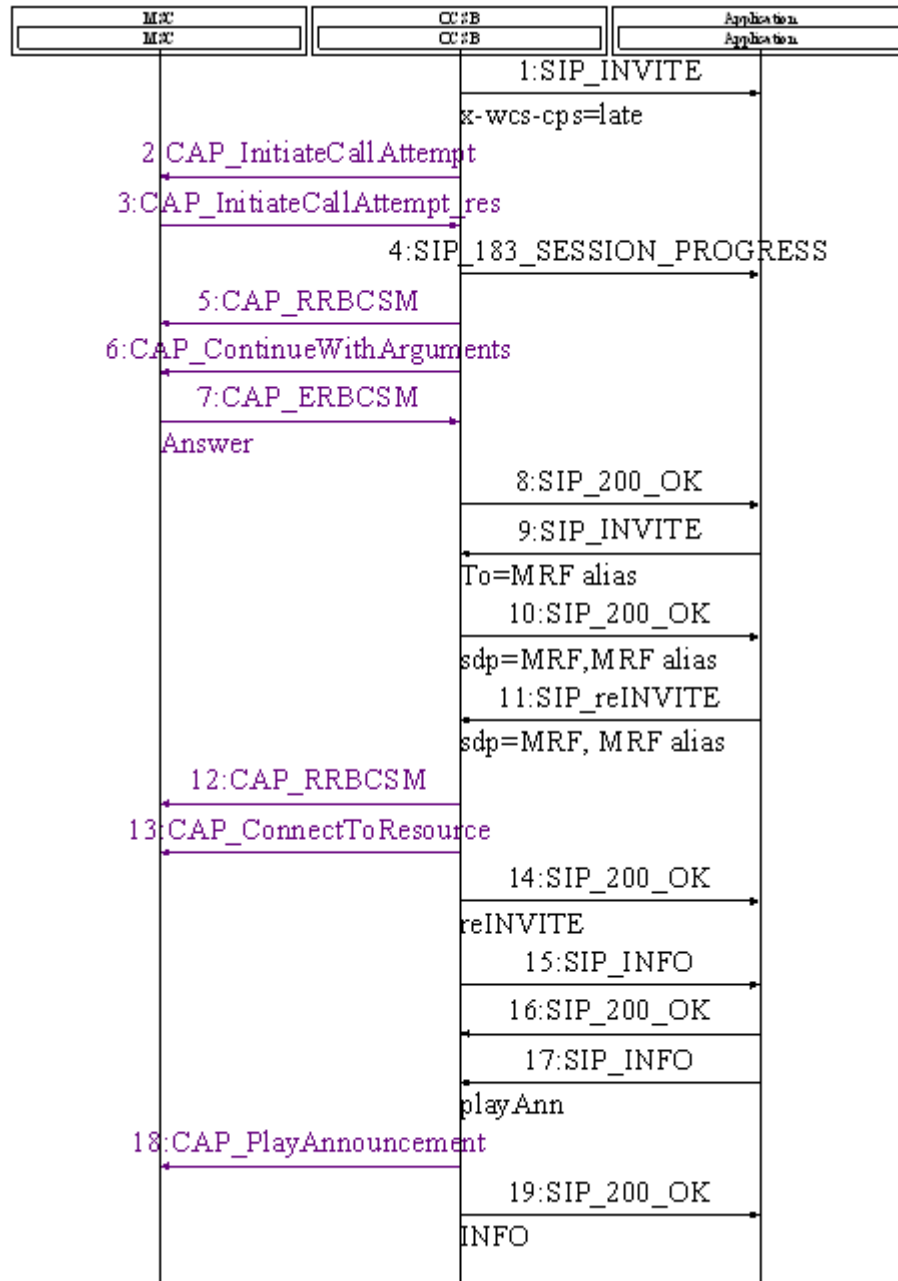
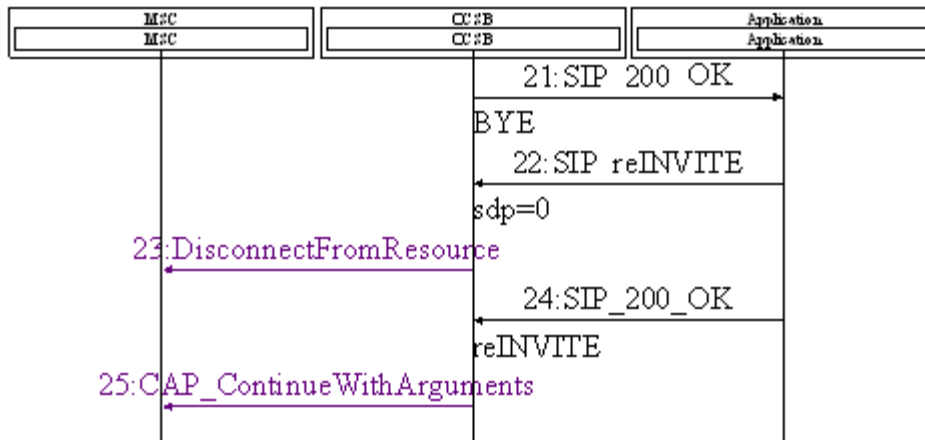


Figure 4-5 Instructing Service Broker to Suspend Call Processing (Cont'd)



You can use this call processing suspension mechanism with both internal and external media resources. However, you can use the call suspension only for network-initiated legs.

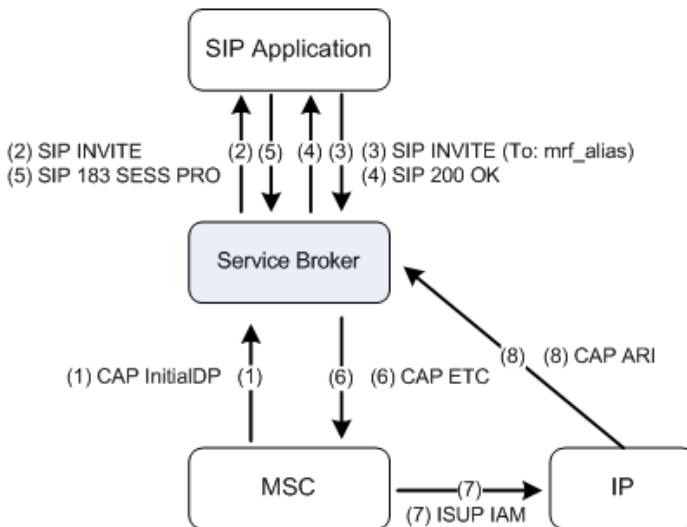
Using an External Media Resource

To use an external media resource, the application sets the To header in the INVITE sent to Service Broker to an MRF alias for which Service Broker is configured to use an external resource.

To instruct an MSC to use an external media resource for user interaction, Service Broker sends a CAP ETC operation towards the MSC. This operation instructs the MSC to set up an ISUP connection towards an Intelligent Peripheral (IP) node. When the IP is invoked, it sends a CAP AssistRequestInstructions (ARI) operation towards Service Broker.

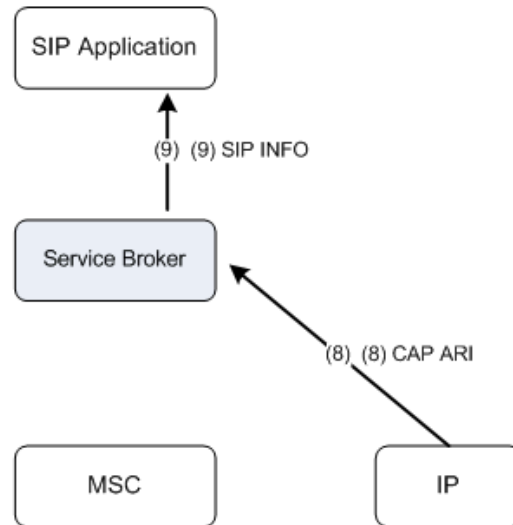
Figure 4-6 and Figure 4-7 show the high level architecture for an application that uses the external resource.

Figure 4-6 Architecture for Sending CAP ETC from Service Broker to the MSC



Service Broker receives the CAP ARI and sends a SIP INFO message towards the application. The INFO message is sent through the dialog created by application. Service Broker attaches a XER representation of the CAP ARI to the SIP INFO body.

Figure 4–7 Architecture for Sending SIP INFO from Service Broker to the Application (CAP ARI)



At this point, the application may use one of the following forms of user interaction:

- Play announcement
- Collect information

The following sections provide information on each of these forms of user interaction.

Playing an Announcement

To play an announcement, the application receives a SIP INFO that contains the CAP ARI, acknowledges the SIP INFO with SIP 200 OK, and sends a new SIP INFO message towards Service Broker. The application sends SIP INFO through the SIP dialog created by the application. The SIP INFO message contains a XER representation of a CAP PlayAnnouncement operation.

Service Broker receives the SIP INFO message and sends a CAP PlayAnnouncement operation towards IP. At this point, the IP plays an announcement to the call party.

[Figure 4–8](#) and [Figure 4–9](#) show the high level architecture for an application that uses an external resource for playing announcement.

Figure 4–8 Architecture for Sending SIP INFO from the Application to Service Broker (CAP PA)

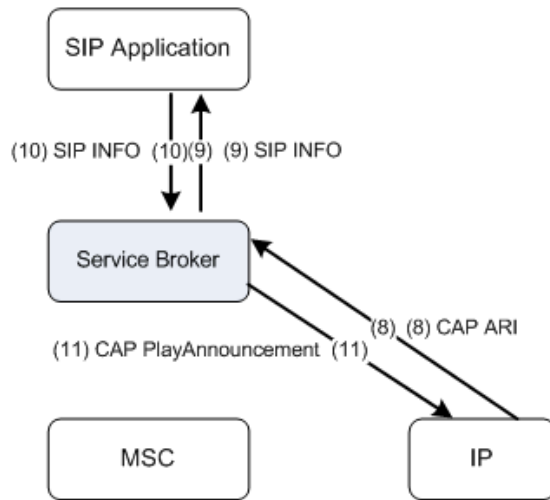
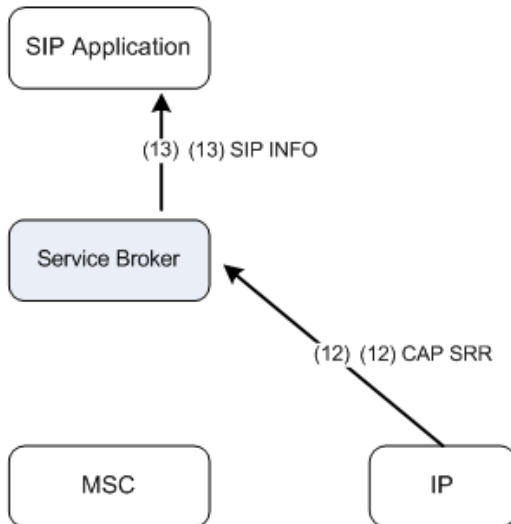


Figure 4–9 Architecture for Sending SDR from Service Broker to the Application (CAP SRR)



The following example shows a XER representation of CAP phase 4 PlayAnnouncement. In this example, the message ID is 6000.

```
<Cap4>
<playAnnouncement>
  <informationToSend>
    <inbandInfo>
      <messageID>
        <elementaryMessageID>6000
      </elementaryMessageID>
      </messageID>
    </inbandInfo>
  </informationToSend>
</playAnnouncement>
</Cap4>
```

When the announcement playing is completed, the IP sends a CAP SpecializedResourceReport (SRR) operation towards Service Broker. Service Broker then sends SIP INFO towards the application through the SIP dialog created by the application. The INFO contains a XER representation of the CAP SpecializedResourceReport operation.

The following example shows a XER representation of CAP phase 4 SpecializedResourceReport.

```
<Cap4>
  <specializedResourceReport>
    <allAnnouncementsComplete/>
  </specializedResourceReport>
</Cap4>
```

At this point, the application can take one of the following actions:

- Continuing the user interaction session by playing additional announcement. The application performs this action by repeating the process of sending a SIP INFO message that contains a CAP PlayAnnouncement XER representation as described in "[Playing an Announcement](#)".
- Terminating the user interaction session, when the application terminates the user interaction session and then either allows the call to continue or rejects the call

Terminating a User Interaction Session

To terminate a user interaction session, the application takes the following actions:

- Sending a SIP BYE request through the dialog created by the application (this is the MRF dialog).
- Updating the SDP towards the initiating side. This is done by sending a SIP 183 SESSION PROGRESS or a SIP reINVITE towards the initiating side. The SIP 183 SESSION PROGRESS or SIP reINVITE contain SDP with the "i" attribute set to "0".

Upon receiving the SIP 183 SESSION PROGRESS or SIP reINVITE message, Service Broker sends a CAP DisconnectForwardConnection (DFC) operation towards the MSC.

[Figure 4–10](#) shows the high level architecture for an application that terminates user interaction session.

Figure 4–10 Architecture for Terminating a User Interaction Session by the Application

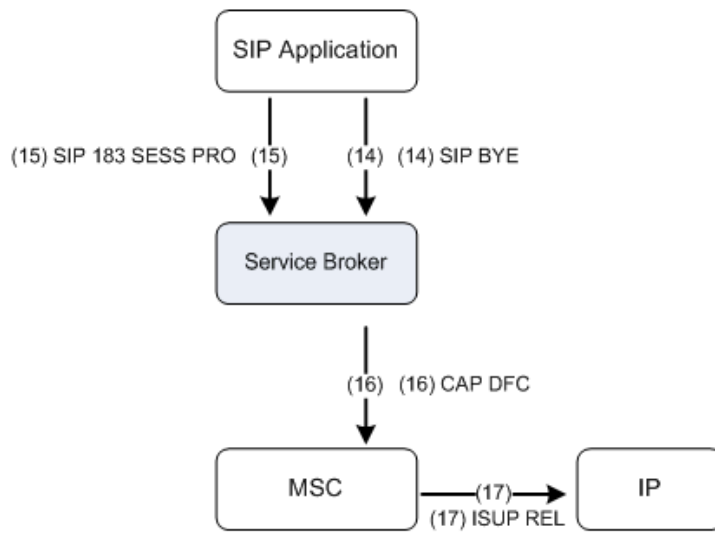


Figure 4–11 and Figure 4–12 show the detailed sequence diagram of the application that uses an external media resource for playing announcement.

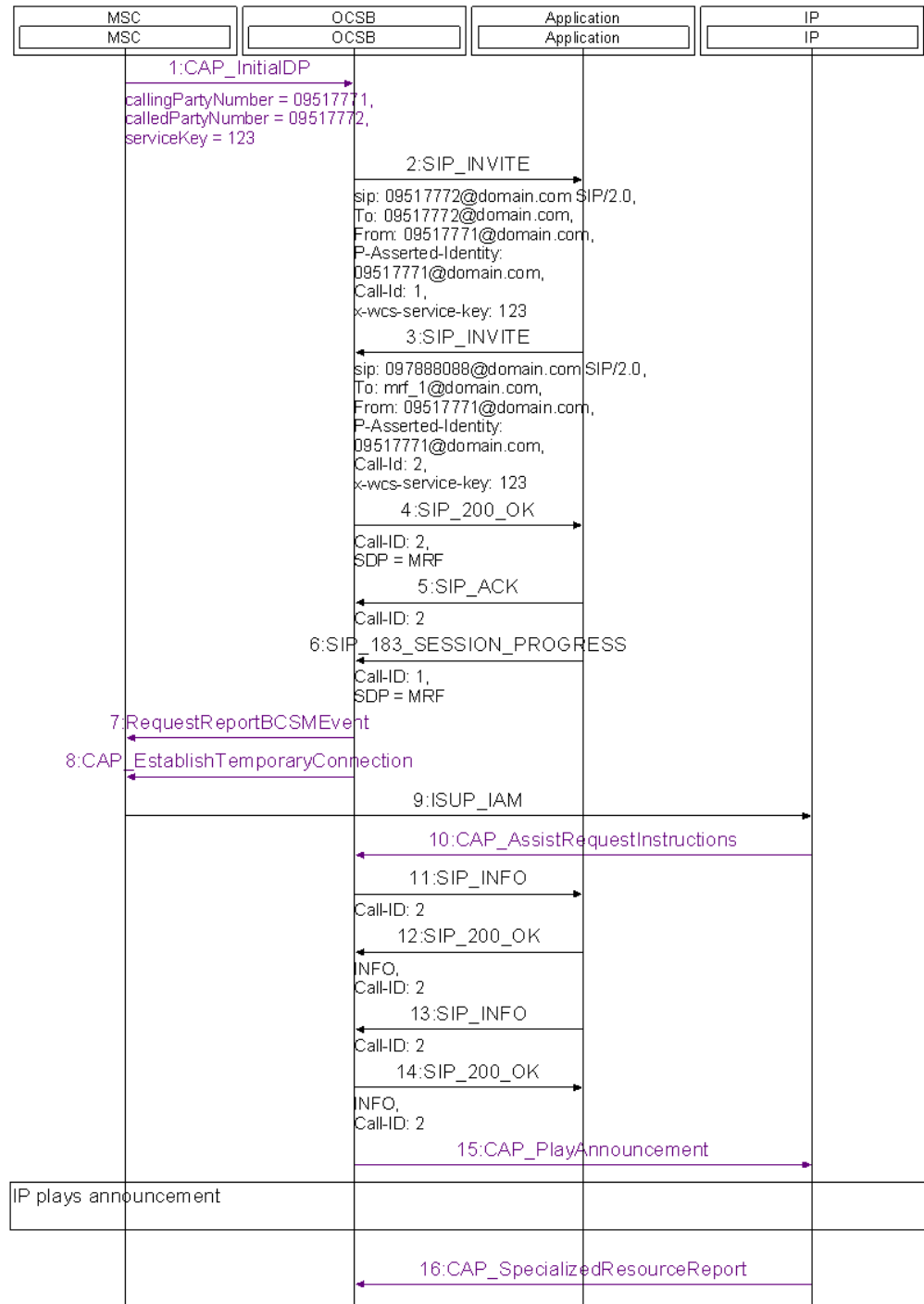
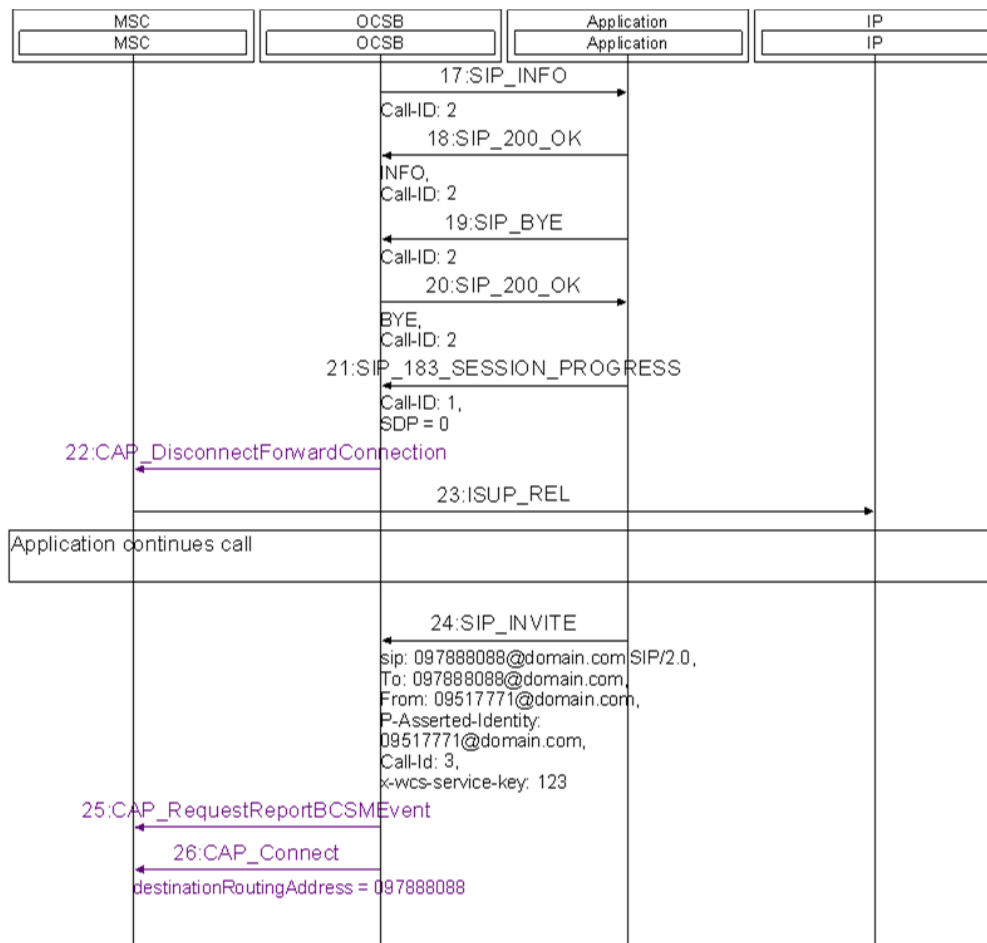
Figure 4–11 Application Uses External Resource for Playing Announcements

Figure 4–12 Application Uses External Resource for Playing Announcement (cont'd)



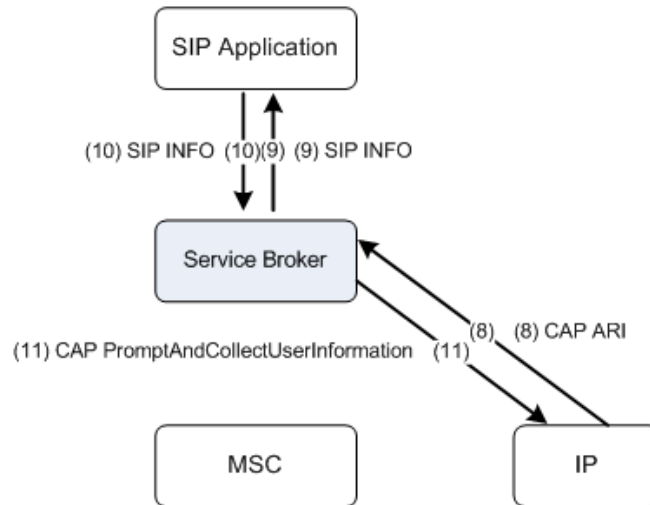
Playing an Announcement and Collecting Information

To play an announcement and collect information, the application receives the SIP INFO which contains the CAP ARI, acknowledges the SIP INFO with SIP 200 OK, and sends a new SIP INFO message towards Service Broker. SIP INFO is sent through the SIP dialog created by the application and contains a XER representation of a CAP PromptAndCollectUserInformation (PnC) operation.

Service Broker receives the SIP INFO and sends a CAP PromptAndCollectUserInformation operation towards the IP. At this point, the IP plays the announcement to the call party and collects user information (DTMF digits).

Figure 4–13 and Figure 4–14 show the high level architecture for an application that uses an external resource for playing announcement and collecting information.

Figure 4–13 Architecture for Sending SIP INFO from the Application to Service Broker (CAP PnC)



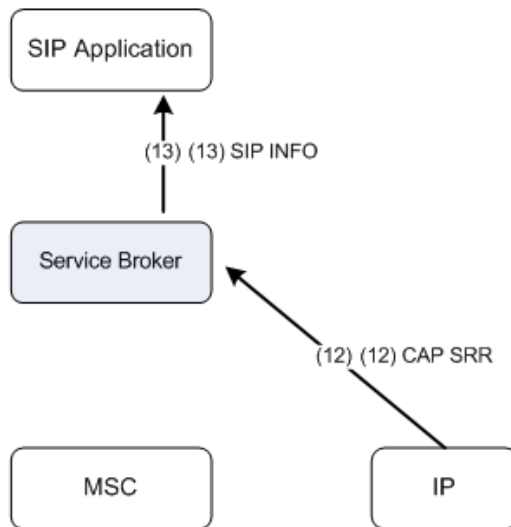
The following example shows a XER representation of CAP PromptAndCollectUserInformation

```
<Cap4>
<promptAndCollectUserInformation>
  <collectedInfo>
    <collectedDigits>
      <maximumNbOfDigits>1</maximumNbOfDigits>
      <firstDigitTimeOut>15</firstDigitTimeOut>
      <interDigitTimeOut>5</interDigitTimeOut>
    </collectedDigits>
  </collectedInfo>
  <informationToSend>
    <inbandInfo>
      <messageID>
        <elementaryMessageID>6023</elementaryMessageID>
      </messageID>
    </inbandInfo>
  </informationToSend>
  <callSegmentID>01</callSegmentID>
</promptAndCollectUserInformation>
</Cap4>
```

When the user interaction is started, that is an announcement is played, the IP sends a CAP SpecializedResourceReport (SRR) operation towards Service Broker. Service Broker then sends a SIP INFO message towards the application through the SIP dialog created by application. The INFO message contains a XER representation of CAP SpecializedResourceReport operation.

Note: IP sends SRR only when this operation is requested in the PromptAndCollectUserInformation operation using the RequestAnnouncementStartedNotification set to "true".

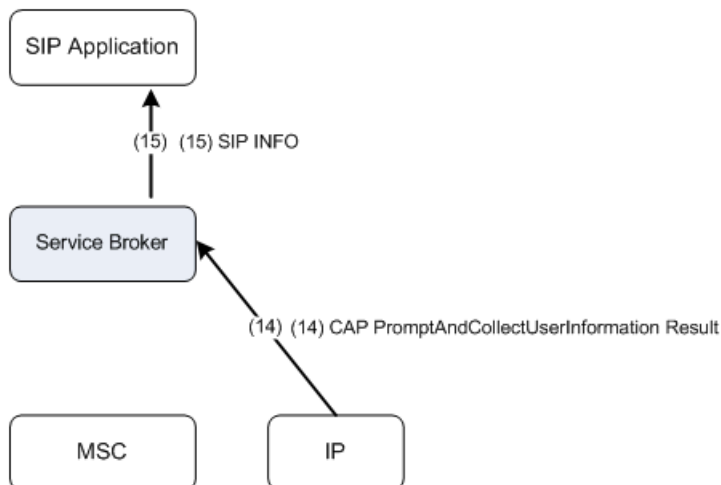
Figure 4–14 Architecture for Sending SIP INFO from Service Broker to the Application (CAP SRR)



When playing the announcement and collecting user information is completed, the IP sends a CAP PromptAndCollectUserInformation result operation that contains the information collected from the user. Then Service Broker sends a SIP INFO message towards the application through the SIP dialog created by the application. The INFO message contains a XER representation of CAP PromptAndCollectUserInformation result operation.

Figure 4–15 shows the high level architecture for an application receiving user information.

Figure 4–15 Architecture for Sending SIP INFO to the Application (CAP PnC Result)



The following example shows a XER representation of CAP PromptAndCollectUserInformation result.

```
<Cap4>
  <promptAndCollectUserInformationResult>
    <digitsResponse>2001</digitsResponse>
  </promptAndCollectUserInformationResult>
</Cap4>
```

At this point, the application can perform one of the following actions:

- To continue the user interaction session by playing additional announcement with or without information collection. The application performs this action by repeating the procedure of sending a SIP INFO message containing a CAP PlayAnnouncement or PromptAndCollectUserInformation XER representation as described in ["Playing an Announcement"](#) and ["Playing an Announcement and Collecting Information"](#).
- To terminate the user interaction session. The application performs this action by terminating the user interaction session. Then the application either allows the call to continue or rejects the call.

[Figure 4–16](#) and [Figure 4–17](#) show the detailed sequence diagram for an application that uses an external media resource for collecting information.

Figure 4-16 Application Uses an External Resource for Collecting Information

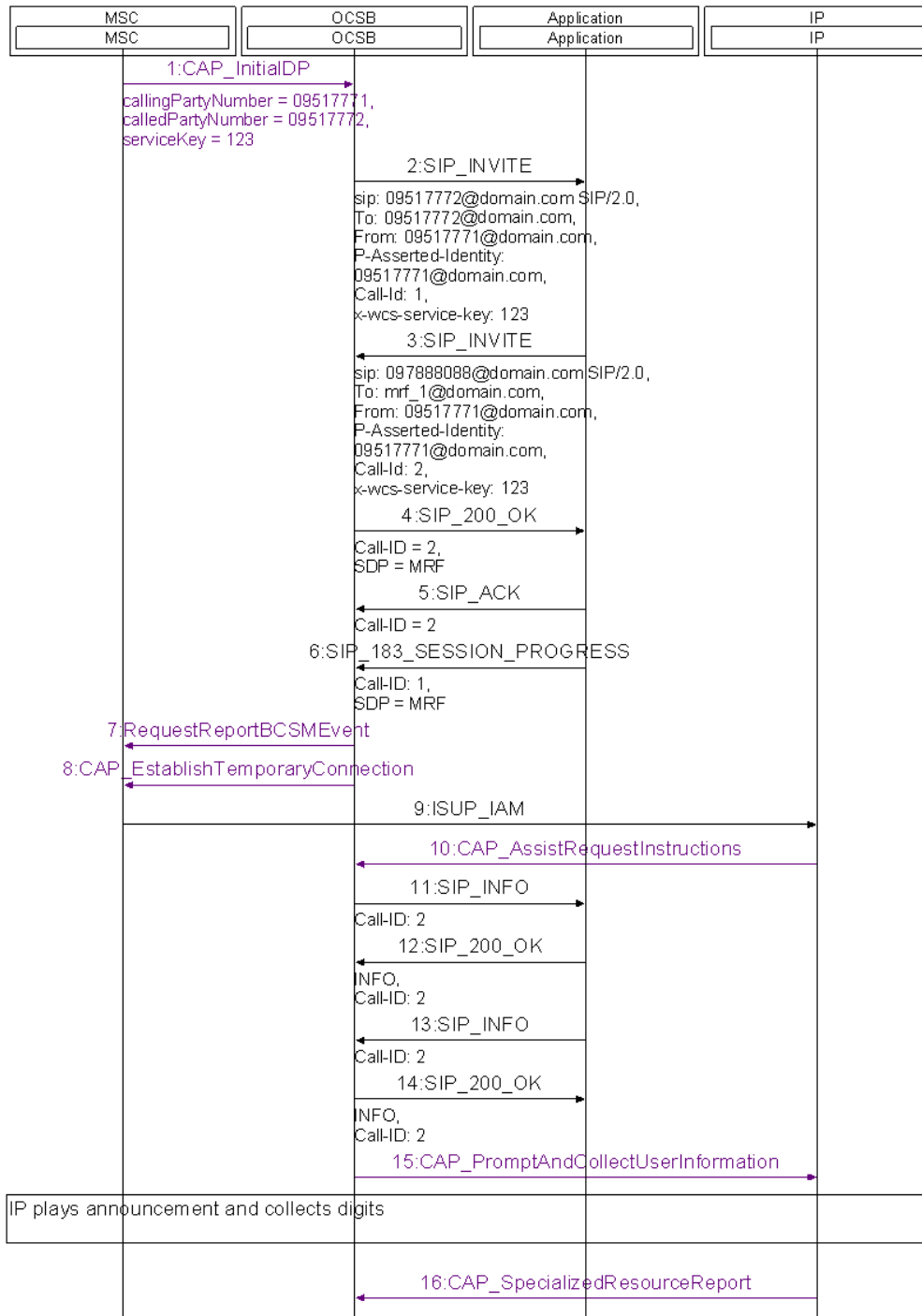
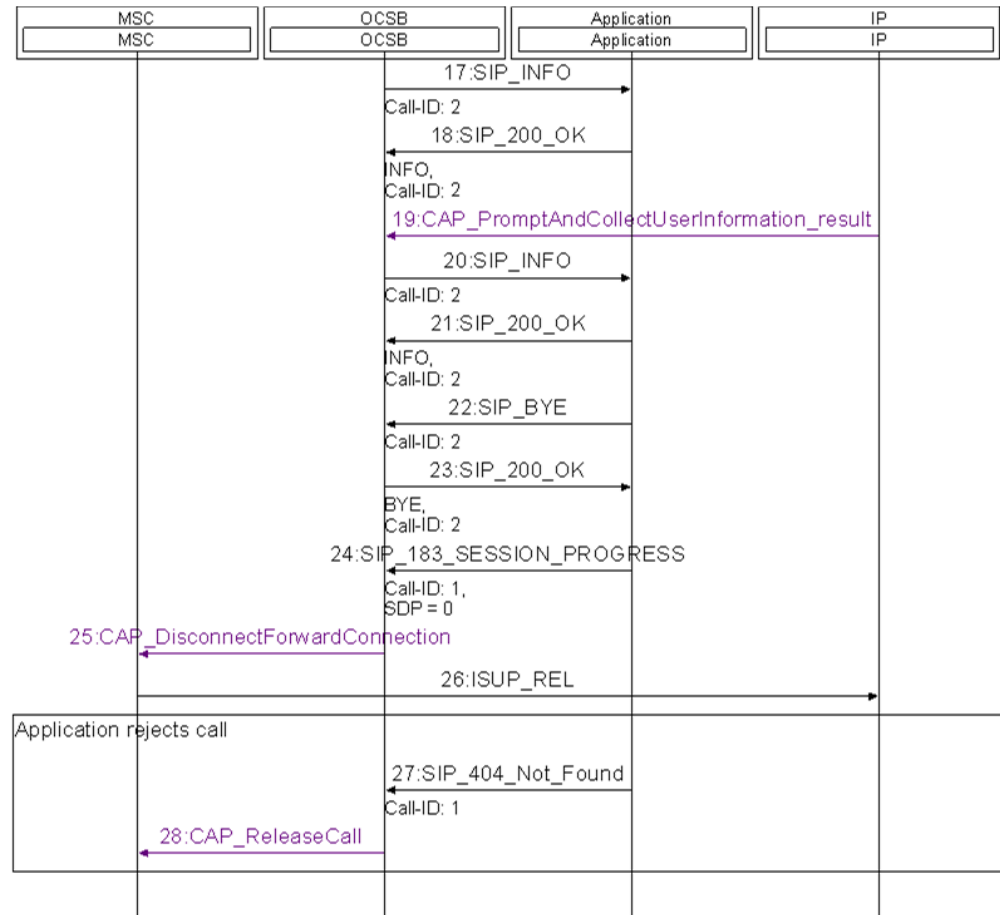


Figure 4–17 Application Uses an External Resource for Collecting Information (cont'd)

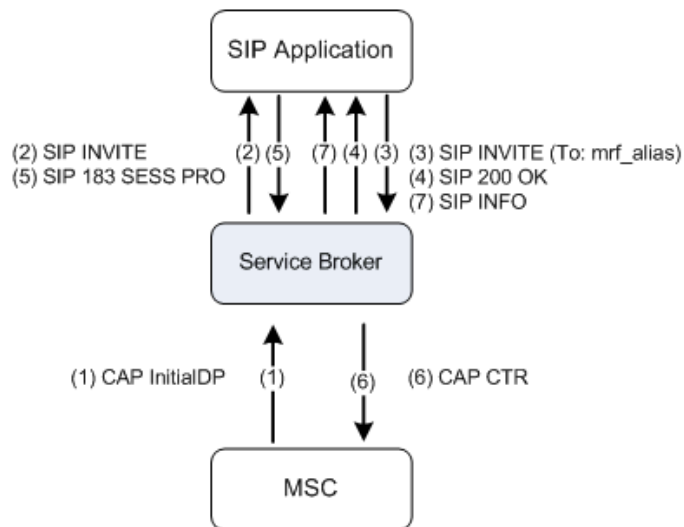
Using an Internal Media Resource

To use an internal media resource, the application sets the To header in the INVITE sent to Service Broker to an MRF alias for which Service Broker is configured to use an internal resource.

To instruct the MSC to use an internal media resource for user interaction, Service Broker sends a CAP ConnectToResource operation towards the MSC.

Upon sending the CAP ConnectToResource, Service Broker sends a SIP INFO message towards the application. The INFO message is sent through the dialog created by the application and does not include an SDP.

Figure 4–18 shows the high level architecture for an application that uses an internal media resource.

Figure 4–18 Architecture for Sending CAP CTR from Service Broker to the MSC

At this point, the application may use one of the following types of user interaction:

- Play announcement
- Collect information

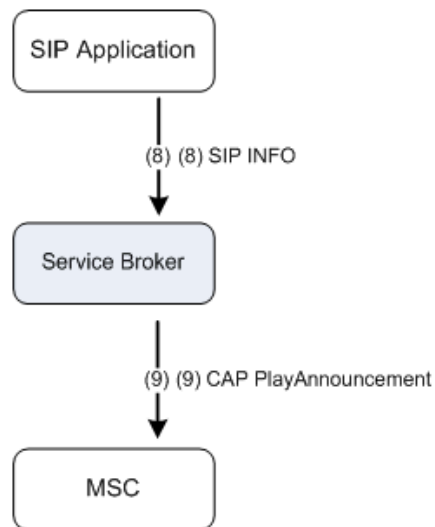
The following sections provide information on each of these forms of user interaction.

Playing an Announcement

To play an announcement, the application receives a SIP INFO message, acknowledges the SIP INFO with SIP 200 OK, and sends a new SIP INFO message towards Service Broker. The SIP INFO message is sent through the SIP dialog created by the application and contains a XER representation of the CAP PlayAnnouncement operation.

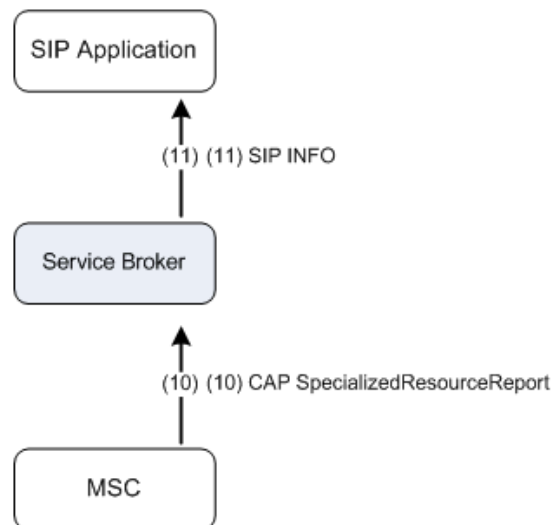
Service Broker receives the SIP INFO message and sends a CAP PlayAnnouncement operation towards the MSC. At this point, the MSC plays an announcement to the call party.

[Figure 4–19](#) shows the high level architecture for an application that uses an internal media resource for playing announcements.

Figure 4–19 Architecture for Using an Internal for Playing Announcements

For the example of a XER representation of CAP phase 4 PlayAnnouncement, see ["Playing an Announcement"](#).

When the announcement playing is completed, the MSC sends a CAP SpecializedResourceReport (SRR) message towards Service Broker. Then Service Broker sends a SIP INFO message towards the application through the SIP dialog created by the application. The INFO message contains a XER representation of the CAP SpecializedResourceReport operation. [Figure 4–20](#) shows the architecture for sending an SRR from Service Broker to an application.

Figure 4–20 Architecture for Sending SRR from Service Broker to the Application

At this point, the application can perform one of the following actions:

- Continuing the user interaction session by playing additional announcement. The application performs this action by repeating the process of sending a SIP INFO message that contains a CAP PlayAnnouncement XER representation as described in ["Playing an Announcement"](#).

- Terminating the user interaction session, when the application terminates the user interaction session and then either allows the call to continue or rejects the call.

Figure 4–21, Figure 4–22, and Figure 4–23 show the detailed sequence diagram for an application that uses an internal media resource for playing announcement.

Figure 4–21 Application Uses an Internal Resource to Play Announcement

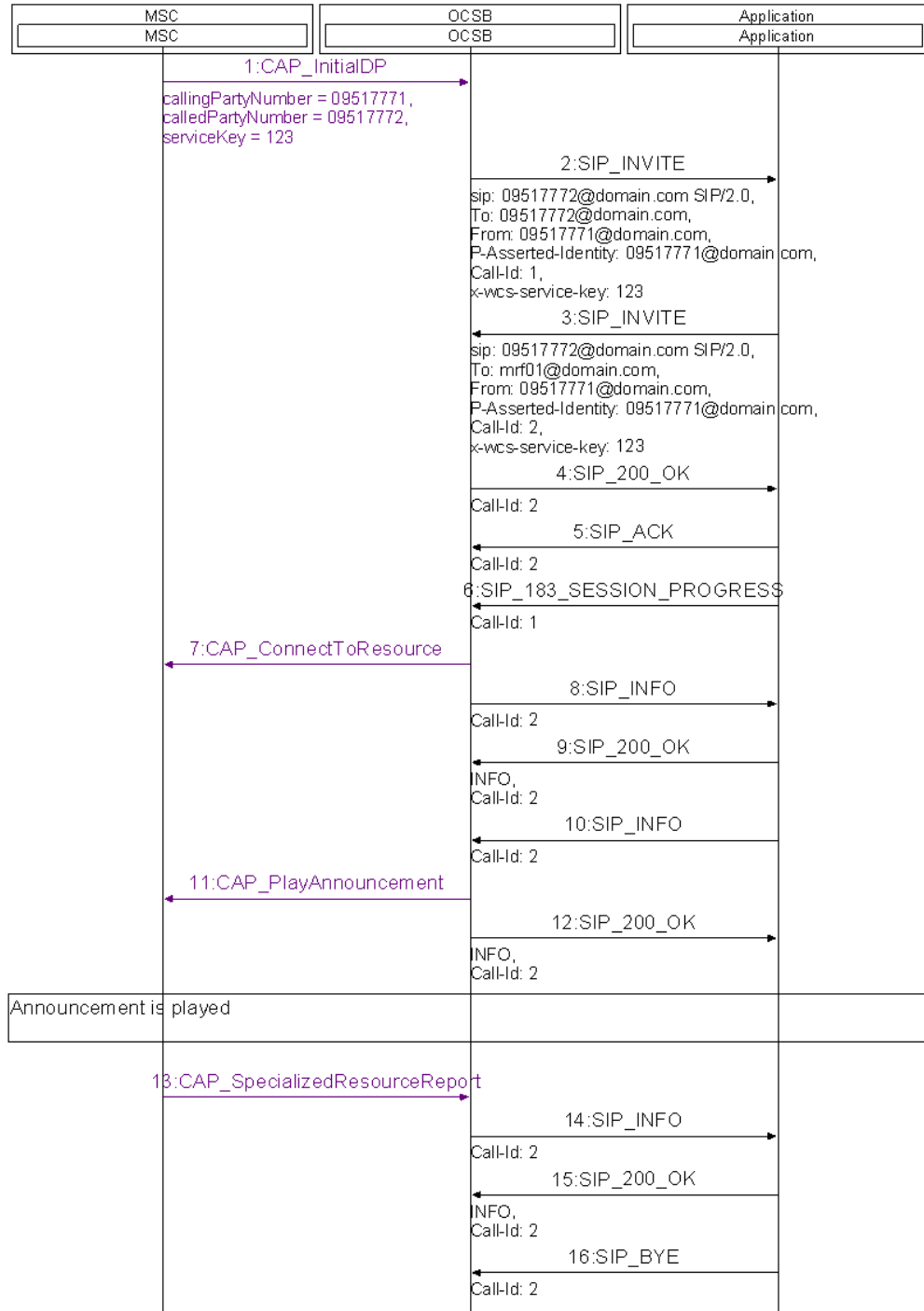
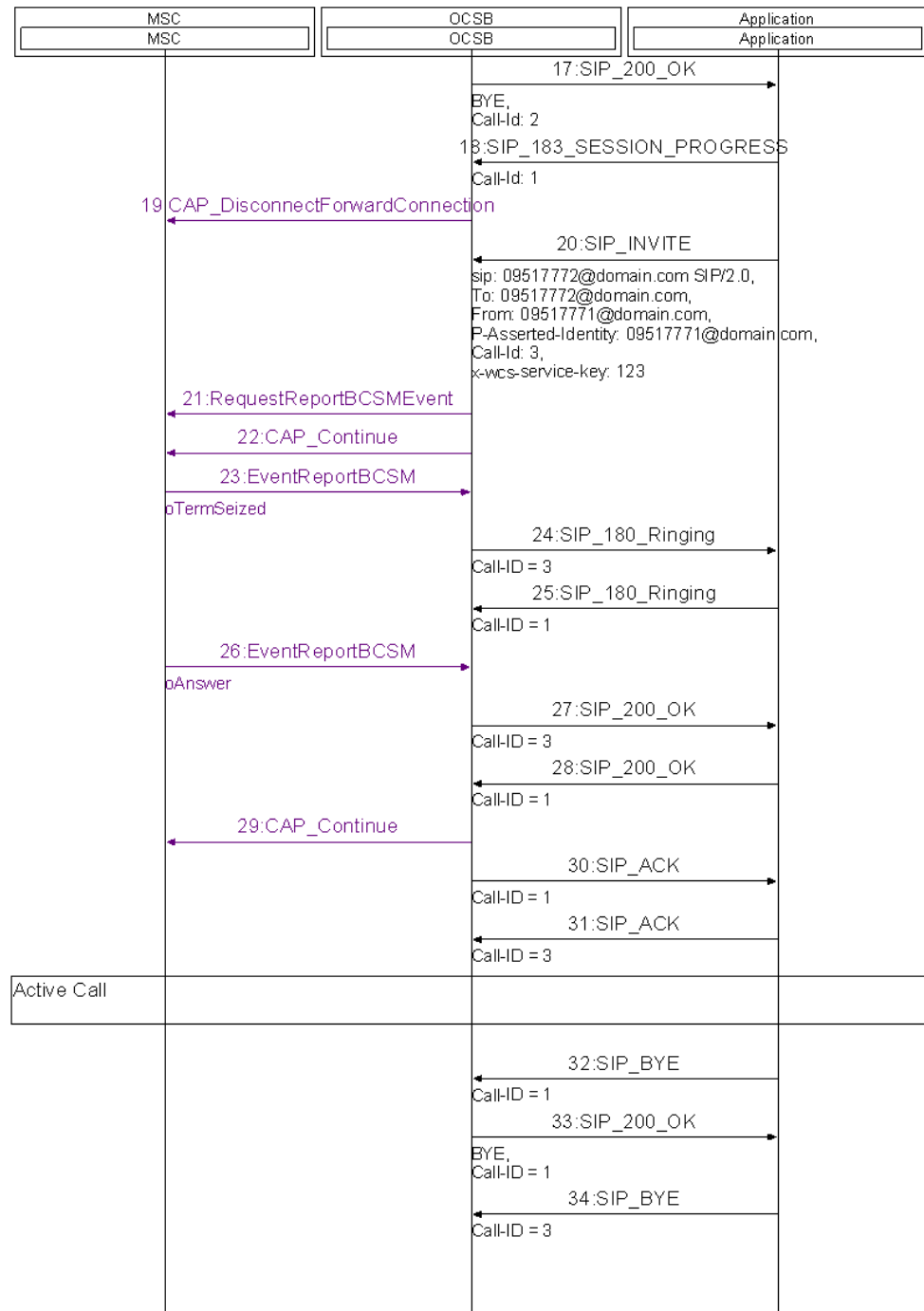
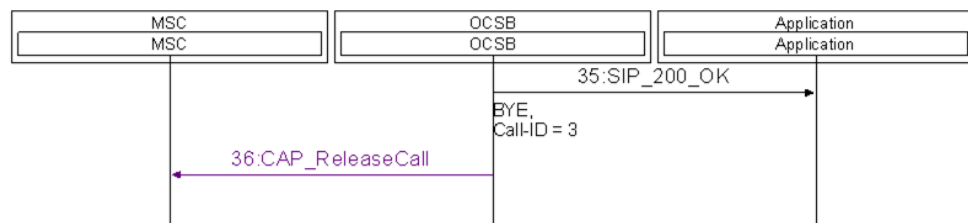


Figure 4–22 Application Uses an Internal Resource to Play Announcement (cont'd)**Figure 4–23 Application Uses an Internal Resource to Play Announcement (cont'd)**

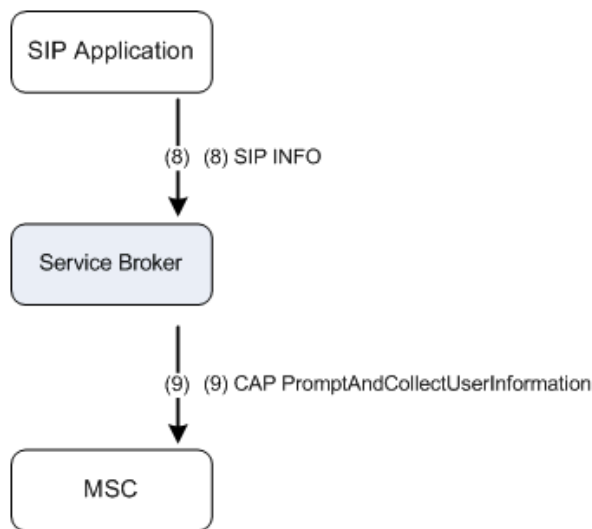
Playing an Announcement and Collecting Information

To play an announcement and collect information, the application receives the SIP INFO, acknowledges the SIP INFO with SIP 200 OK, and sends a new SIP INFO message towards Service Broker. The SIP INFO message is sent through the SIP dialog created by the application and contains a XER representation of the CAP PromptAndCollectUserInformation (PnC) operation.

Service Broker receives the SIP INFO and sends a CAP PromptAndCollectUserInformation operation towards the MSC. At this point, the MSC plays the announcement to the call party and collects user information (DTMF digits).

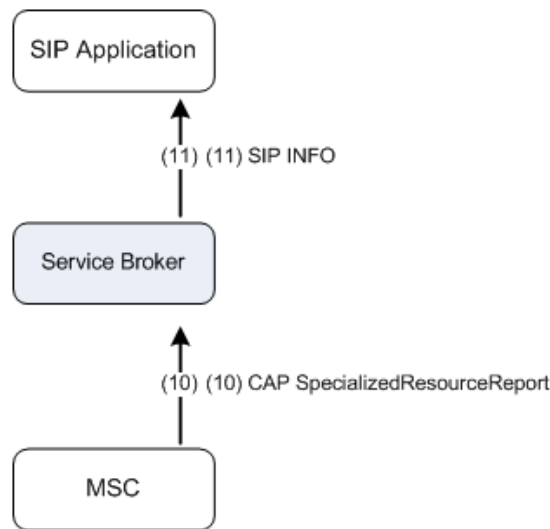
Figure 4–24 shows the high level architecture for an application that uses an external resource for playing announcement and collecting information.

Figure 4–24 Architecture for Using an External Resource for Playing Announcements



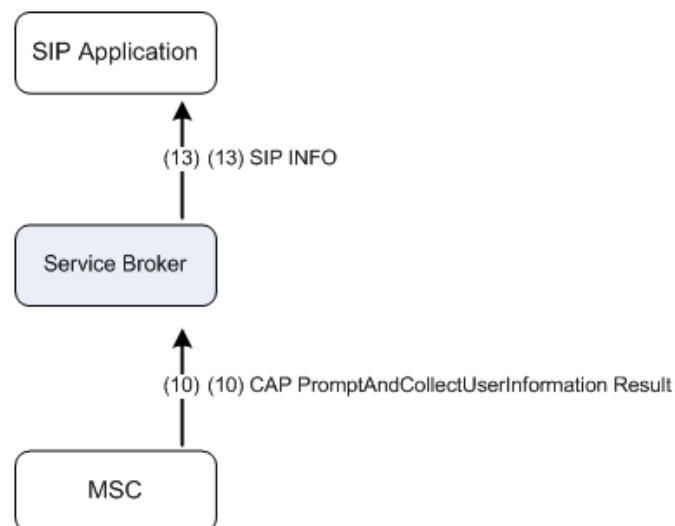
For the example of a XER representation of the CAP phase 4 PromptAndCollectUserInformation, see "[Playing an Announcement and Collecting Information](#)".

When user interaction is started, that is an announcement is played, the MSC sends a CAP SpecializedResourceReport (SRR) operation towards Service Broker. Service Broker then sends a SIP INFO towards the application through the SIP dialog created by the application. The INFO message contains a XER representation of the CAP SpecializedResourceReport operation. Figure 4–25 shows the architecture for sending a SIP INFO message from Service Broker to an application.

Figure 4–25 Architecture for Sending SIP INFO from Service Broker to the Application

When playing the announcement and collecting user information is completed, the MSC sends a CAP PromptAndCollectUserInformation result operation that contains the information collected from the user. Then Service Broker sends a SIP INFO message towards the application through the SIP dialog created by the application. The INFO message contains a XER representation of the CAP PromptAndCollectUserInformation result operation.

Figure 4–26 shows the high level architecture for an application that receive user information.

Figure 4–26 Architecture for Sending SIP INFO to the Application

For the example of a XER representation of the CAP phase 4 PromptAndCollectUserInformation Result, see "[Playing an Announcement and Collecting Information](#)".

At this point, the application can perform one of the following actions:

- Continuing the user interaction session by playing additional announcement. The application can perform this action by repeating the procedure of sending a SIP INFO message that contains a CAP PromptAndCollectUserInformation XER

representation as described in ["Playing an Announcement and Collecting Information"](#).

- Terminating the user interaction session. The application performs this action by terminating the user interaction session. Then the application either allows the call to continue or rejects the call as described in ["Terminating a User Interaction Session"](#).

[Figure 4–27](#), [Figure 4–28](#), and [Figure 4–29](#) show the detailed sequence diagram for an application that uses an internal media resource for collecting information.

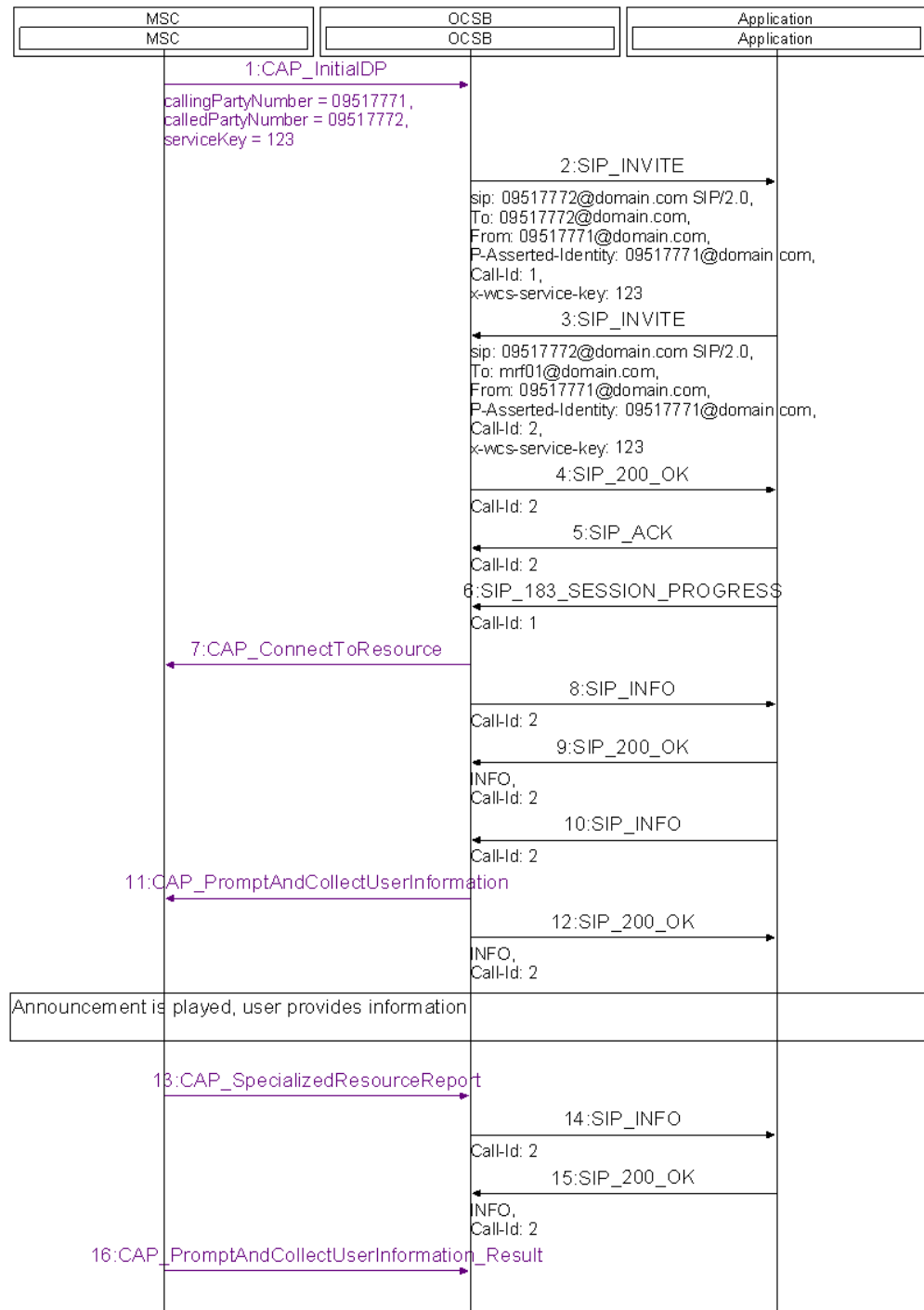
Figure 4-27 Application Uses an Internal Resource for Collecting Information

Figure 4–28 Application Uses an Internal Resource for Collecting Information (cont'd)

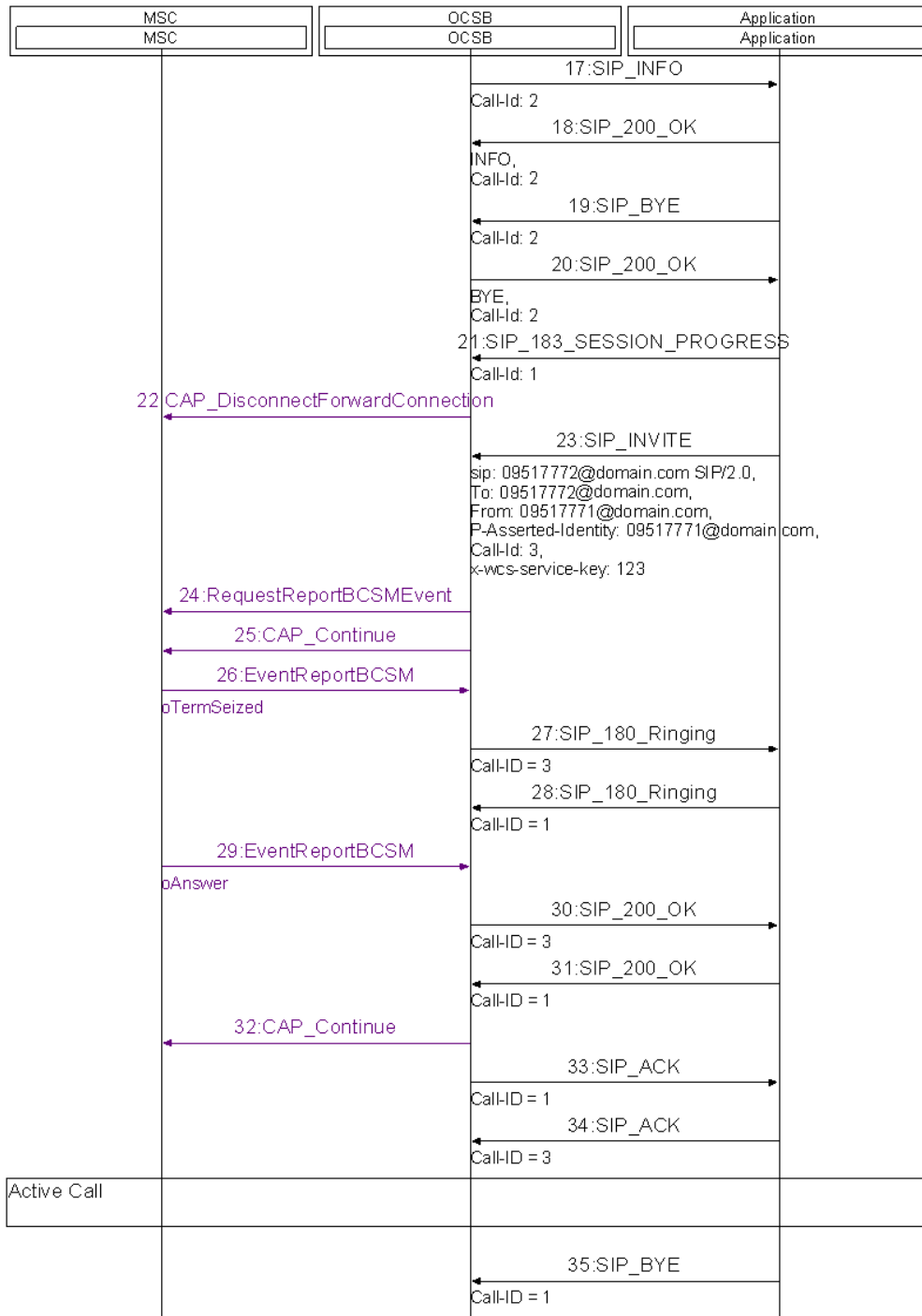
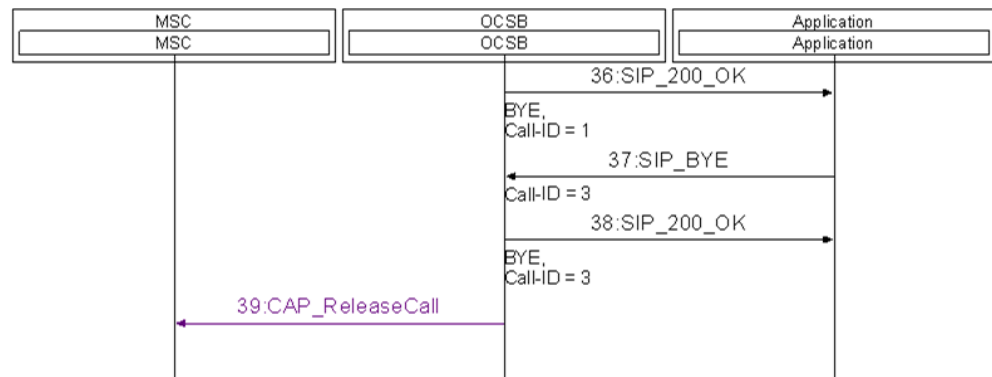


Figure 4–29 Application Uses an Internal Resource for Collecting Information (cont'd)

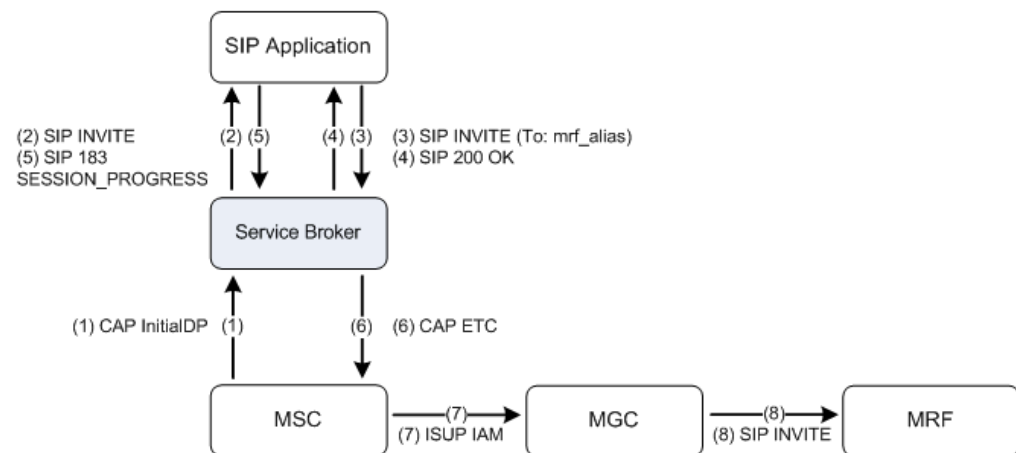
SIP MRF Integration for User Interaction Functionality

Service Broker enables a SIP application to use a SIP MRF to provide user interaction functionality. To do so, the application creates an MRF dialog as described in ["Using an External Media Resource"](#) and ["Using an Internal Media Resource"](#).

Depending on the MRF alias used by the application, Service Broker sends a CAP EstablishTemporaryConnection or CAP ConnectToResource towards the MSC. The EstablishTemporaryConnection or ConnectToResource is set with a destination number as configured in the Service Broker's Media resources table.

The MSC sends an ISUP IAM towards a Media Gateway Controller (MGC) that creates a SIP dialog towards the SIP MRF.

[Figure 4–30](#) shows the high level architecture for MRF integration.

Figure 4–30 Architecture for MRF Integration

From this point, the MRF exchanges information (for example, voiceXML) with the application using a dedicated interface (for example, HTTP).

Using this method, the application should correlate the HTTP session with the SIP MRF dialog created towards Service Broker. This is done using the alias used by the application and propagated by Service Broker through the MSC and MGC towards the MRF.

[Figure 4–31](#) and [Figure 4–32](#) show the detailed sequence diagram of an application that uses a SIP MRF for user interaction.

Figure 4-31 Application Uses a SIP MRF for User Interaction

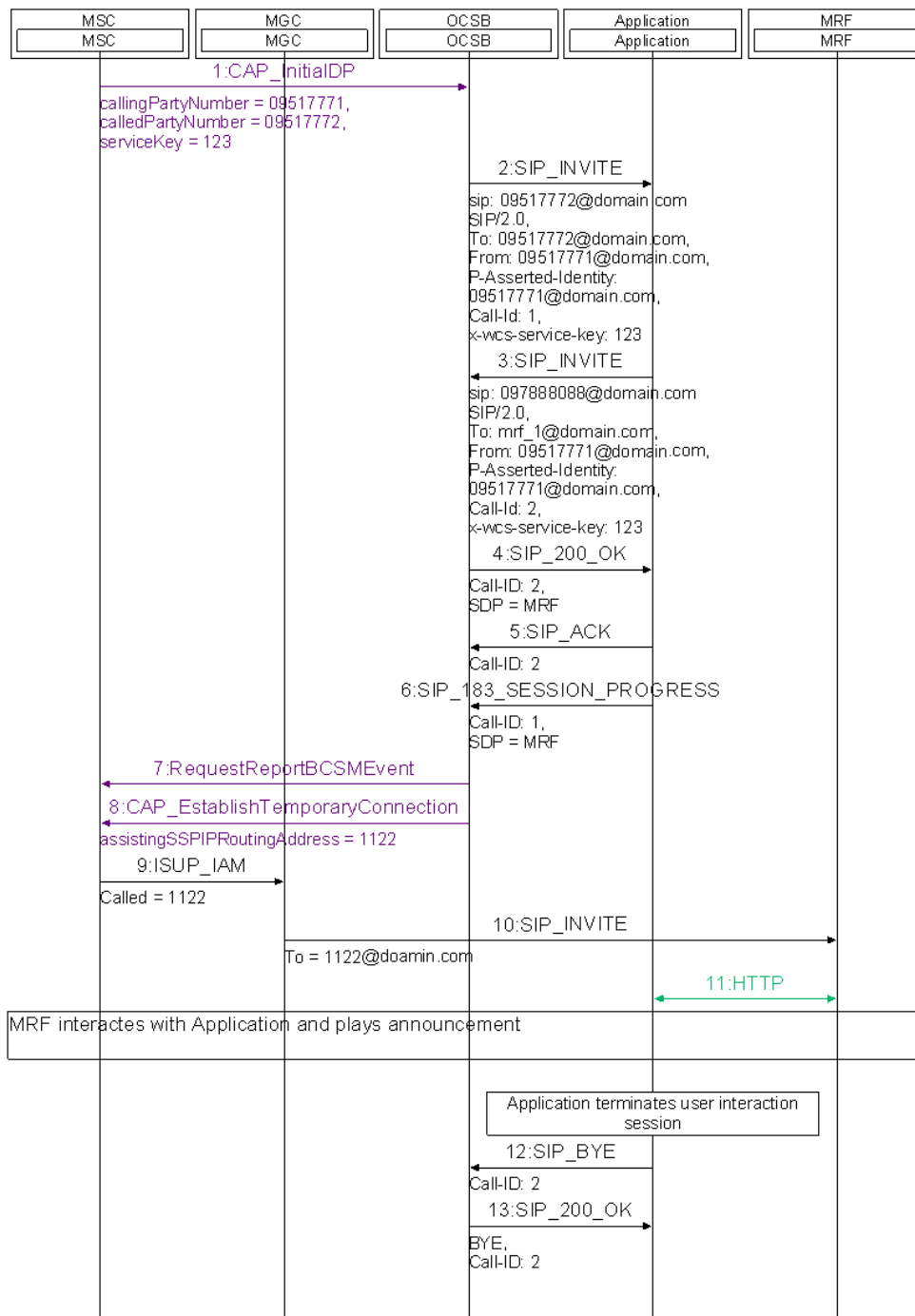
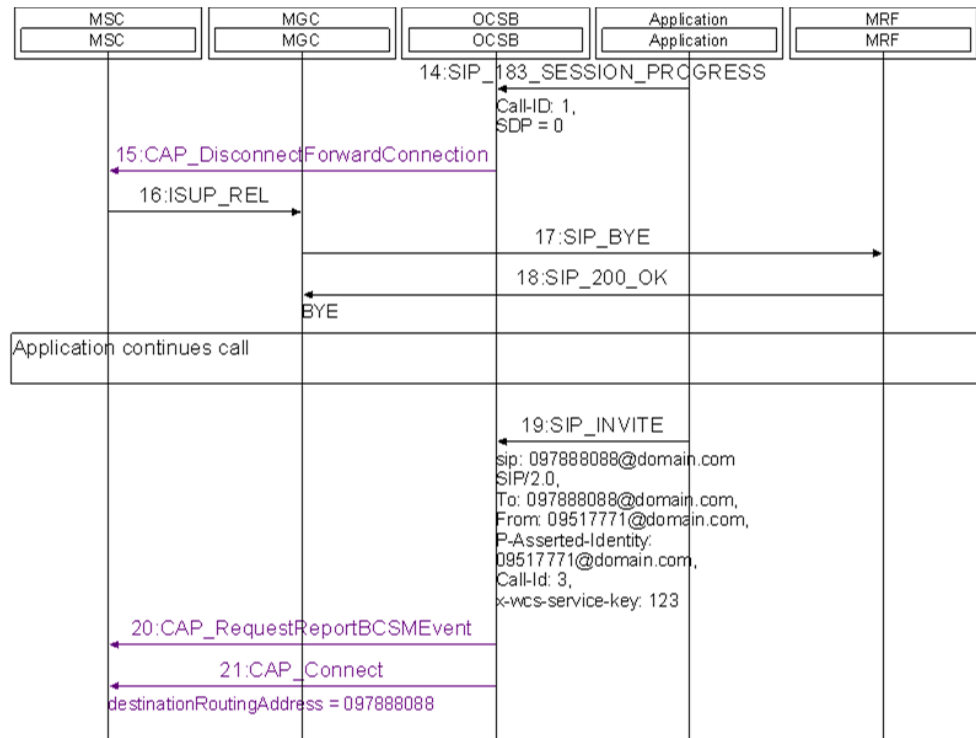


Figure 4-32 Application Uses a SIP MRF for User Interaction (cont'd)



Developing a Multi-Leg Call Control Application

This chapter describes how to develop a SIP multi-leg call control application.

Providing Multi-Leg Call Control

Multi-leg call control is the method of having an application controlling a call that involves two or more call parties. This method is used by enhanced applications to provide services, such as personalized ring-back tone and call transfer.

The multi-leg call control functionality enables an application to control individual parties in a call by performing the following actions:

- Creating a new leg, when the application creates a new leg in a new call or creates an additional leg in an existing call
- Disconnecting leg, when the application disconnects a specific leg
- Removing a leg from a call, when the application removes a leg from a call and placing that leg on hold
- Connecting call legs, when the application connects two or more call legs.

Service Broker enables the application to control a multi-leg call in accordance with SIP third part call control conventions. These conventions relay on SDP negotiation across call legs as described in the following sections.

About CAP Call Leg Representation

Service Broker uses the SDP carried by SIP messages to represent CAP call leg information. Specifically, the “i” attribute of SDP is used for this purpose as shown in the following examples:

[Figure 5–1](#) shows an example in which Service Broker sets a SIP INVITE sent to application with SDP of i=l1. l1 represents leg number 1, that is the call initiating leg.

Figure 5–1 Architecture for Setting the SDP to leg 1

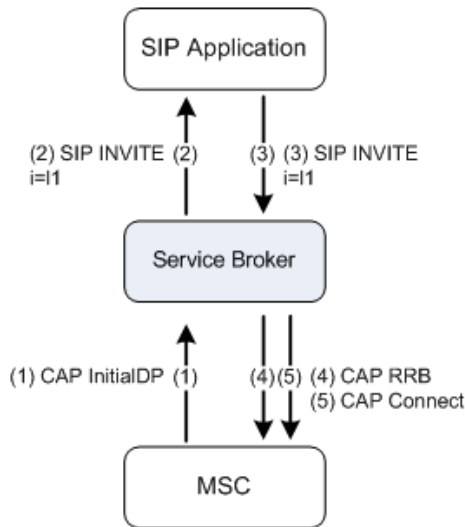
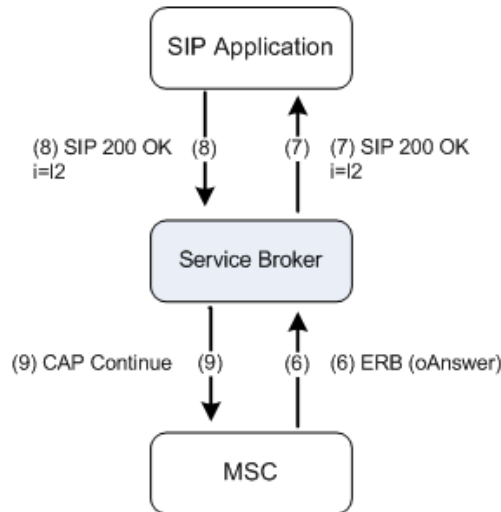


Figure 5–2 shows an example in which Service Broker sets a SIP 200 OK which acknowledges INVITE with SDP of i=12. 12 represents leg number 2, that is the call destination leg.

Figure 5–2 Architecture for Setting the SDP to leg 2



As shown on Figure 5–1 and Figure 5–2, each leg created Service Broker on the CAP interface is represented by Service Broker using the “i” attribute of the SDP. Service Broker sets the “i” attribute to the number assigned by Service Broker to that leg.

Creating a New Call Leg

An application can create a new call leg in one of the following forms:

- Creating a new leg in a new call
- Creating a new leg in an existing call

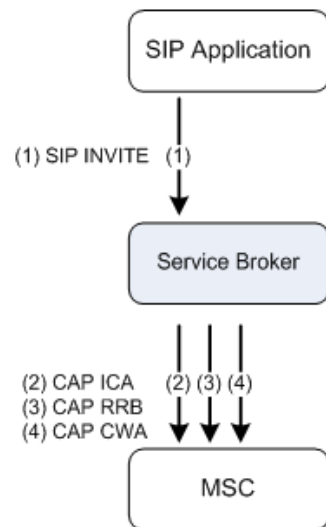
Creating a New Leg in a New Call

To create a new leg in a new call, the application sends a SIP INVITE message to Service Broker. The application sets the INVITE with an empty SDP, that is the INVITE message does not include SDP.

When the SIP INVITE message is received, Service Broker sends a CAP InitiateCallAttempt (ICA) operation towards the MSC. The InitiateCallAttempt is accompanied by a CAP RequestReportBCSM and ContinueWithArgument operations.

Figure 5–3 shows the high level architecture for an application creating a new call leg.

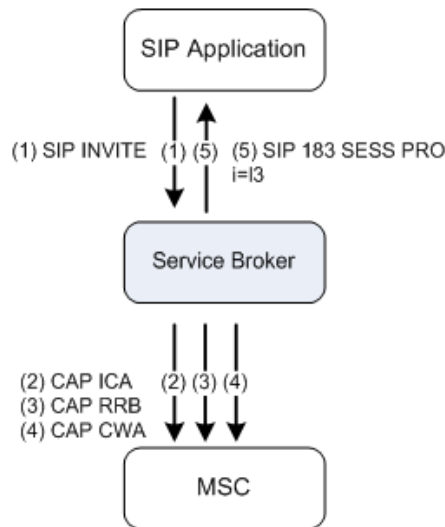
Figure 5–3 Architecture for Creating a New Call Leg



When the CAP InitiateCallAttempt is sent, Service Broker sends a SIP 183 SESSION PROGRESS message to the application. Service Broker sets the 183 SESSION PROGRESS with SDP of $i=l(x)$ where x is the number assigned by Service Broker to the newly created leg.

Figure 5–4 shows the high level architecture for an application that receives the call leg assigned by Service Broker to a new leg.

Figure 5–4 Architecture for Sending SIP 183 SESSION PROGRESS from Service Broker to the Application



Creating a New Leg in an Existing Call

To create a new leg in an existing call, the application follows the procedure described in ["Creating a New Leg in a New Call"](#).

To enable Service Broker to associate the new leg with an existing call, the application sets the SIP INVITE message which is used to create the new leg, with a Route header (one or more) and includes the same content as in the header of the SIP INVITE used to create previous legs in that call.

Restricting Calling Line Identification for a Calling Party Number

You restrict the calling line identification by setting the **Presentation** indicator to **"presentation restricted"**. You can restrict the calling line identification in one of the following ways:

- On a permanent basis by setting the **Presentation** indicator to **"presentation restricted"** for all ICA messages that Service Broker generates.
- On a per-message basis by setting the **Presentation** indicator to **"presentation restricted"** for specific ICA messages.

Restricting Calling Line Identification on a Permanent Basis

If you want to set the **Presentation** indicator to **"presentation restricted"** in all ICA messages that Service Broker generates, in the sip.xml file of the relevant IM-SCF module, set the **string_ address-presentation-restriction-indicator-of-calling-party-number-of-ICA** parameter to **restricted**.

Restricting Calling Line Identification on a Per-Message Basis

If you want to set the **Presentation** indicator to **"presentation restricted"** for specific ICA messages only, add the following XER to the ICA that Service Broker generates:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Cap4>
```



```

<initiateCallAttempt>
  <destinationRoutingAddress>
    <OCTET_STRING>8490630975071008</OCTET_STRING>
  </destinationRoutingAddress>
  <callingPartyNumber>8213 0A00 000000</callingPartyNumber>
</initiateCallAttempt>
</Cap4>

```

The first 4 digits of the callingPartyNumber are setting the numbering format, including the presentation and are specified in ITU-T Q763 section 3-10

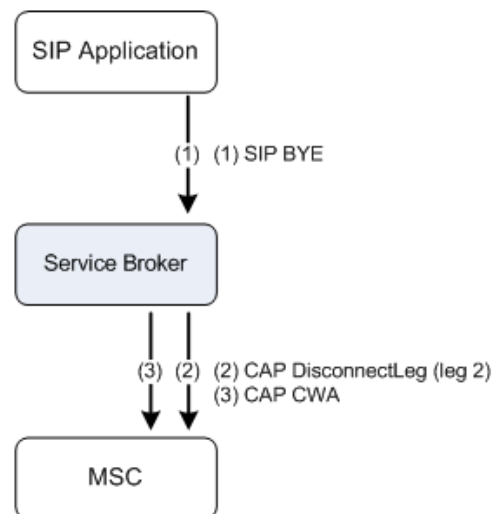
Disconnecting a Call Leg

To disconnect a leg, the application sends a SIP BYE or a SIP CANCEL request. The SIP BYE or SIP CANCEL request is sent through the SIP dialog that represents the leg that the application wants to disconnect.

When Service Broker receives a SIP BYE or a SIP CANCEL, Service Broker waits for a predefined configurable period of time. If during this period, all call legs are terminated by the application (that is, the application sends a SIP BYE and or SIP Cancel for all call legs), Service Broker terminates the call by sending a CAP ReleaseCall. Otherwise, (that is the application does not terminate all call legs), Service Broker sends a CAP DisconnectLeg accompanied by a CAP ContinueWithArgument towards the MSC for each leg terminated by the application.

Figure 5–5 shows the high level architecture for an application that disconnects the leg. Figure 5–5 assumes that two call legs (1 and 2) exist, and the call is active.

Figure 5–5 Architecture for Disconnecting the Leg



Removing a Leg from a Call

To remove a leg from a call, the application sends a SIP reINVITE, or a SIP 183 SESSION PROGRESS message depending on the actual call state.

When the SIP reINVITE or SIP 183 SESSION PROGRESS message is received, Service Broker sends a CAP SplitLeg operation towards the MSC followed by CAP ContinueWithArgument.

Connecting Call Legs

To connect a leg to another existing leg, the application sends a SIP reINVITE, SIP 183 SESSION PROGRESS or SIP 200 OK message depending on the actual call state.

When the SIP reINVITE, SIP 183 SESSION PROGRESS message or SIP 200 OK, is received, Service Broker sends a CAP MoveLeg operation towards the MSC followed by CAP ContinueWithArgument.

Multi-Leg Control Example

This section provides an example of a multi-leg call control application.

[Figure 5-6](#), [Figure 5-7](#), and [Figure 5-8](#) show the detailed sequence diagram for an application that controls a multi-leg call. The call is established as an ordinary 2 legs call. When the call is active, the application creates a new call leg (leg 3).

Note: To create a new call leg, the application sends a SIP INVITE (no SDP) message and keeps the Route header with the same content as the Route header of the SIP INVITE sent by the application during the call establishment phase (message 3 in the sequence diagram).

While leg 3 is being alerted, Service Broker sends a SIP 180 Ringing to the application. At this point, the application removes leg 2 from the call and puts it on hold. This is done by sending a SIP reINVITE message on the dialog created by Service Broker. The application sets the SDP of the reINVITE to leg 3. Latter on, when leg 1 and leg 3 are in an active call, the application disconnects leg 3 by sending a SIP BYE request on the corresponding SIP dialog and than reconnects leg 1 and leg 2.

Figure 5-6 Application Provides Multi-Leg Call Control

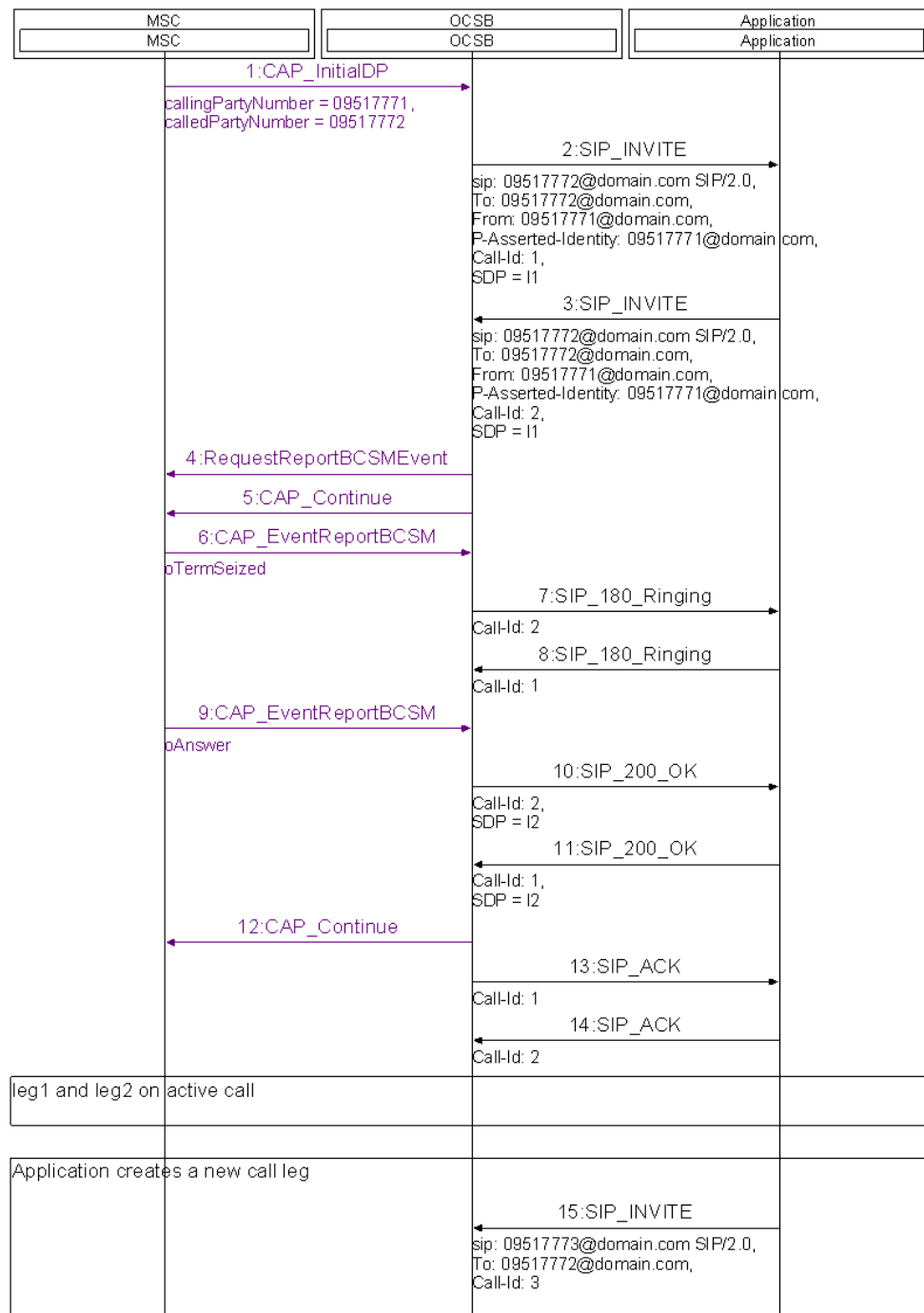


Figure 5-7 Application Provides Multi-Leg Call Control (cont'd)

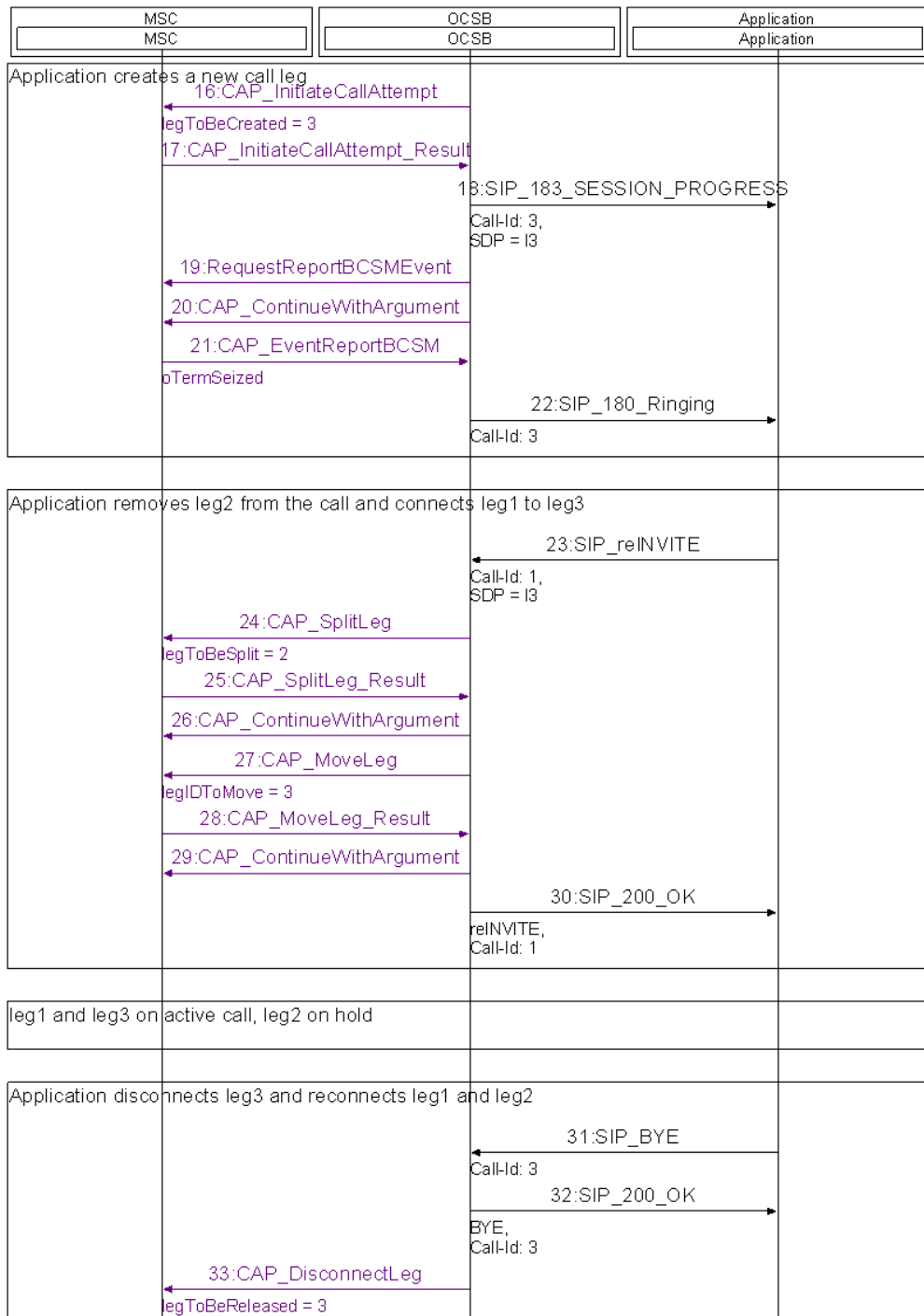


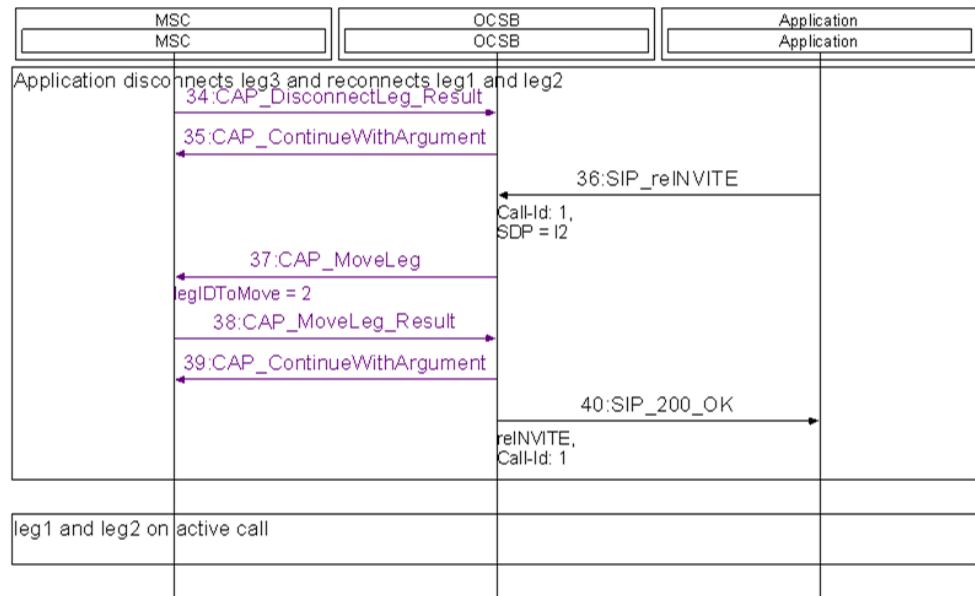
Figure 5–8 Application Provides Multi-Leg Call Control (cont'd)

Figure 5–9, Figure 5–10, and Figure 5–11 show a parallel ringing application. This application routes the call to two destinations simultaneously (leg3 and leg 4) by sending two SIP INVITE messages (no SDP). When one of the legs answers the call, the application connects that leg to leg 1 by sending the SDP of the answering leg towards leg 1 using SIP 200 OK message (message 21 in the sequence diagram). Next, the application terminates the other leg (leg 4) by sending SIP CANCEL.

Figure 5–9 Parallel Ringing Application

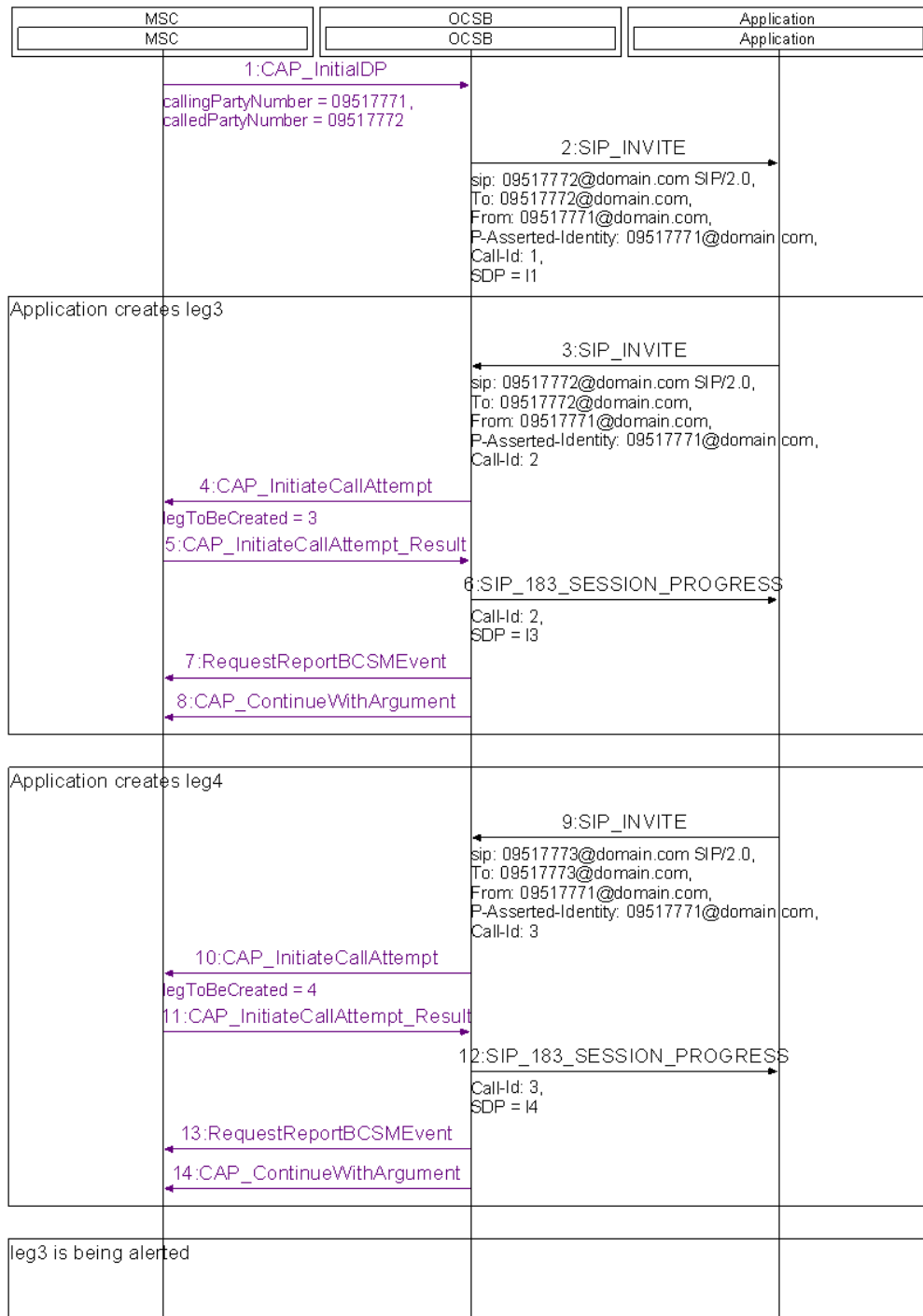


Figure 5–10 Parallel Ringing Application (cont'd)

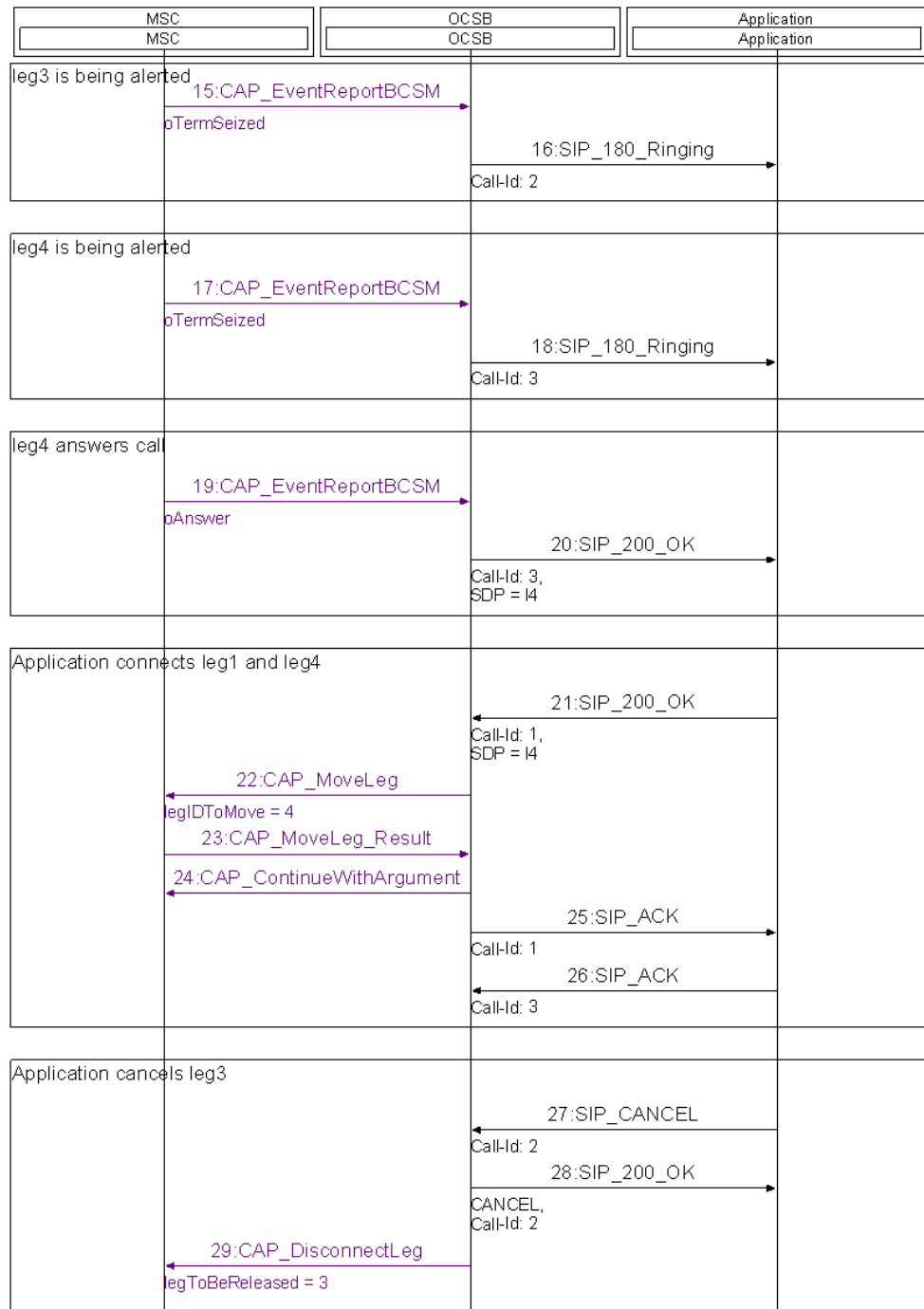
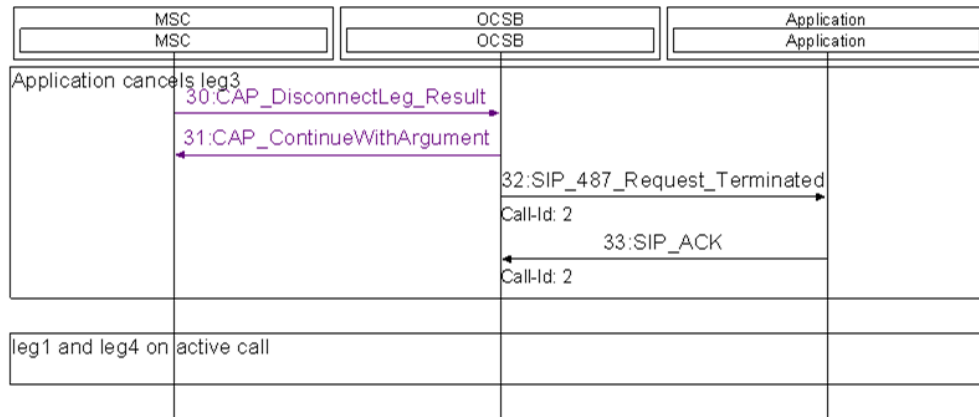


Figure 5-11 Parallel Ringing Application (cont'd)



Understanding the Service Broker NG-IN Solution for Presence and Subscriber Status Applications

This chapter describes the principles of the Oracle Communications Service Broker NG-IN solution for Presence and Subscriber Status eXtensions applications.

Introduction

The Service Broker NG-IN solution enables a SIP application to access SS7 network entities that communicate using the MAP protocol. The solution currently supports interaction with HLRs and VLRs.

With the Service Broker NG-IN solution for PSX applications, your SIP application can access a mobile network to perform the following actions:

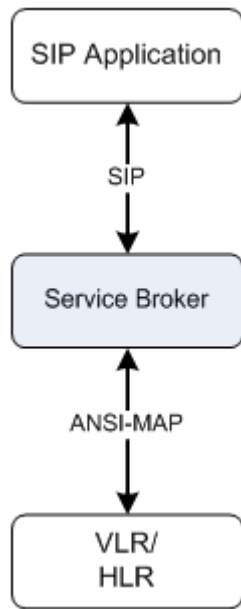
- Obtain a mobile subscriber's state, location, and service subscription information from an HLR
- Modify mobile subscriber's subscription information in an HLR and VLR

Solution Architecture

The Service Broker NG-IN solution for PSX applications consists of the following components:

- One or more SIP applications
- Service Broker
- MAP network entity

[Figure 6-1](#) shows a SIP application that interacts with an HLR or VLR in a legacy mobile network.

Figure 6–1 Architecture for Interacting with MAP Network Entities

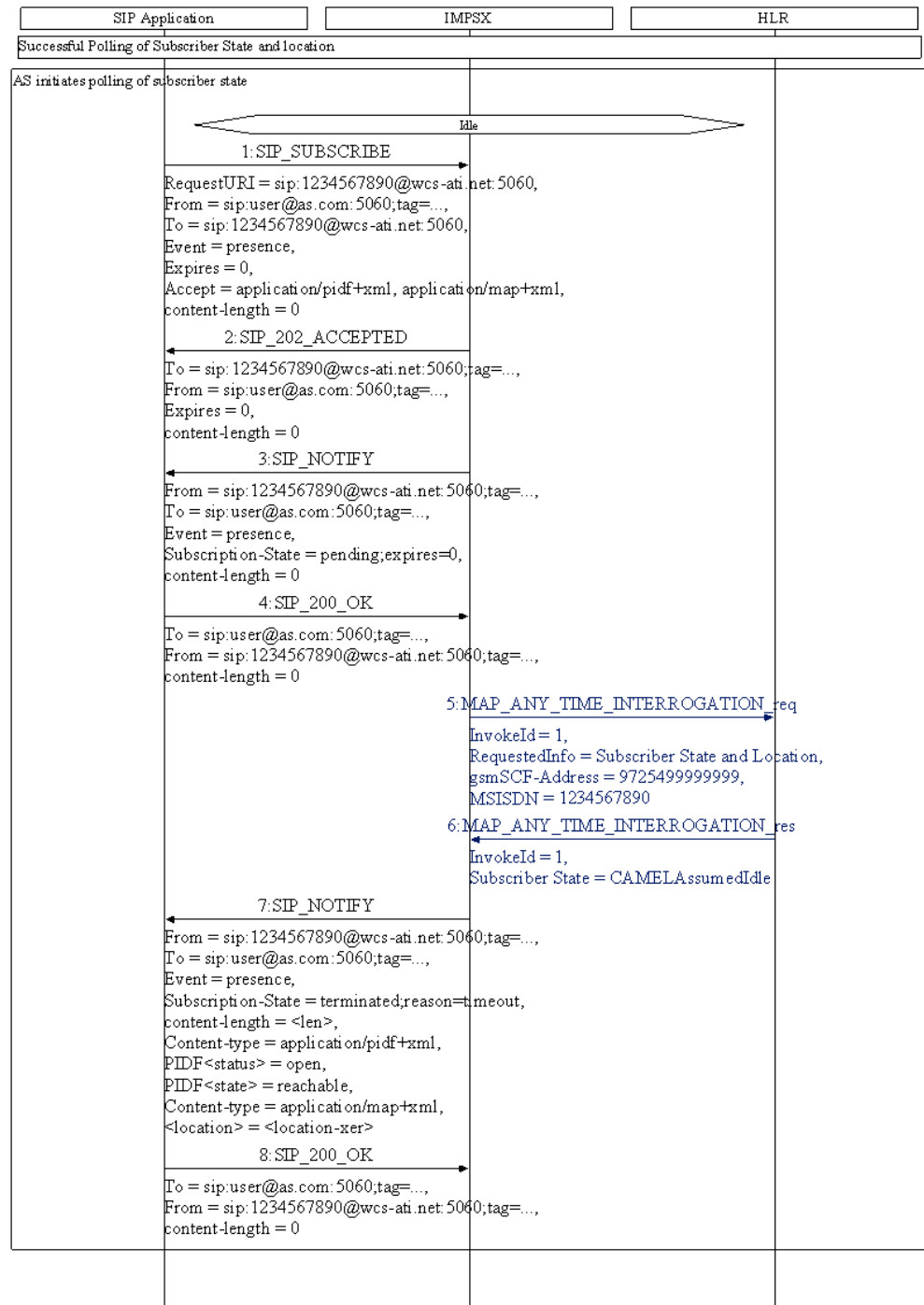
The application-facing side of Service Broker provides a SIP application with a standard SIP interface. The interface is based on the SIP SUBSCRIBE and SIP NOTIFY messages. Implementing the interaction between a SIP application and a network entity does not require any network-specific customization.

SIP SUBSCRIBE and SIP NOTIFY Interface

A SIP application interacts with Service Broker through a standard SIP interface using the subscribe and notify mechanism. [Figure 6–2](#) shows a typical call flow in the solution:

1. The SIP application subscribes to Service Broker for a specific type of operation, such as obtaining the subscriber's status. The application performs the subscription by sending a SIP SUBSCRIBE message to Service Broker.
2. The SIP SUBSCRIBE message triggers Service Broker to perform an appropriate operation on the SS7 network entity.
3. After Service Broker received a response from the entity, Service Broker sends the application either a SIP NOTIFY message or a failure response. In both cases, Service Broker terminates the subscription by
 - Setting the Subscription-State header of the SIP NOTIFY message to "terminated"
 - Setting the "reason" token of the Subscription-State header of the SIP NOTIFY message to "timeout"

Figure 6–2 Basic Interrogation Call Flow



Exchanging Information Through the SIP Interface

Service Broker exchanges information with the SIP application through the common SIP interface using two different mechanisms:

- SIP headers

To provide Service Broker with information required to trigger specific MAP operations, the SIP application uses SIP headers.

Service Broker also supports several SIP header tokens. For example, Service Broker supports the **requested-info** token to allow the SIP application to specify which information it wishes to obtain.

- SIP message body

Service Broker uses the SIP message body to exchange the following two types of information:

- Accept XER or BER encoded MAP operation arguments that need to be sent towards SS7 entities. For example, Service Broker uses the SIP SUBSCRIBE message body to accept MAP ANY-TIME-SUBSCRIPTION-INTERROGATION argument, and further construct the argument before sending it towards the SS7 entity.
- Pass XER or BER encoded MAP operation results to the SIP application.

Developing a Presence and Subscriber Status SIP Application

This chapter describes how to develop a Presence and Subscriber Status eXtensions SIP application in the Oracle Communications Service Broker NG-IN solution.

Understanding Common SIP Interface Concepts

Using the NG-IN solution, SIP applications can obtain mobile subscriber's information that is stored in SS7 network entities such as HLR. To obtain subscriber's information, a SIP application has trigger a SIP SUBSCRIBE message to Service Broker and specify inside the information that it needs. Service Broker returns the subscriber's information in a SIP NOTIFY message.

The following sections describe common parts of the SIP SUBSCRIBE and NOTIFY interface, such as headers and SIP errors, that the SIP application generates every time it communicates with Service Broker, regardless of the type of operation the SIP application requests to perform.

Specifying the Address of an SS7 Entity

SIP applications can instruct Service Broker which SS7 entity to connect using the domain part of the To header. An application can set the domain part using one of the following methods:

- Setting an alias

When a SIP application needs to communicate with an SS7 entity whose SCCP address is configured in the SS7 SSU, the application sets an alias that refers to this address.

Service Broker uses this alias to resolve the preconfigured SCCP address (that is point code or GT address). For example, if you set the To header to **sip:1234567890@h1r01**, then Service Broker will resolve h1r01 to a real SCCP address, based on the SCCP addresses configured in the SS7 SSU.

Note that if you specify an alias that resolves to a dynamic GT address, then you must also specify the GT digits, using the 'gtaddr' token. For example, **sip:1234567890@h1r01;gtaddr=972543349098**

- Using an IM-PSX SIP domain

When a SIP application needs to communicate with an SS7 entity whose alias is preconfigured in IM-PSX, in the **PsxSipDomain** parameter, the application sets the IM-PSX SIP domain.

When you set the IM-PSX SIP domain in the domain part of the To header, for example, **sip:1234567890@ocsb-psx.net**, Service Broker uses the alias configured in the **DefaultSs7EntityAlias** parameter to resolve an SCCP address that is already configured in the SS7 SSU.

- Using the Anonymous string

If a SIP application needs to communicate with an entity whose SS7 address is not configured in Service Broker, the application constructs the domain part of the To header using the "Anonymous" string.

Setting the "Anonymous" string means the application uses additional tokens in the To header to specify an SCCP address.

The "Anonymous" string requires an application to set the tokens described in [Table 7-1](#).

Table 7-1 Tokens Required to Specify an SCCP Address

Token	Type	Description
gtaddr	STRING	Stands for Global Title Address. The parameter contains digits.
nai	STRING	Stands for Nature of Address Indicator. Possible values: <ul style="list-style-type: none"> ■ unknown ■ subscriberNumber ■ nationalReserved ■ nationalSignificant ■ international
np	STRING	Stands for Numbering Plan. Possible values: <ul style="list-style-type: none"> ■ unknown ■ isdn ■ generic ■ data ■ telex ■ maritimeMobile ■ landMobile ■ isdnMobile
netind	STRING	Stands for Network Indicator. Possible values: <ul style="list-style-type: none"> ■ national ■ international
tt	BYTE	Stands for Translation Type.
spc	Up to 24 bit decimal	Stands for Signaling Point Code.
ssn	BYTE	Stands for Subsystem Number.

An application must set the tokens described in [Table 7-1](#) as follows:

- The **netind** token is always required.
- One of the following tokens is required:
 - **gtaddr**. If an application uses the **gtaddr** token, to allow Service Broker to build the global title indicator, the application must set at least one of the following tokens: **tt** or **nai**.
 - **spc** accompanied by the **ssn** token

For example, the following combinations of tokens are considered valid:

- **netind, gtaddr, nai, np, tt**
- **netind, gtaddr, tt**
- **netind, gtaddr, tt, np**
- **netind, gtaddr, nai**
- **netind, spc, ssn**
- **netind, gtaddr, nai, np, tt, spc, ssn**

Specifying the Identity of a Mobile Subscriber

SIP applications specify the mobile subscriber whose information is required by using the domain part of the RequestURI.

SIP applications should set the RequestURI as follows:

- Set the mobile subscriber's MSISDN in the user part
- Set the IM-PSX address, as configured in **PsxSipDomain**, in the domain part.

A SIP application can use the **noa** token, to specify the MSISDN nature of address that is later used on the MAP interface. The possible **noa** token values are:

- subscriber
- unknown
- national

To specify that the MSISDN nature of address is 'international', the SIP application must set the MSISDN in the RequestURI user part in a global format, with a leading '+' sign.

For example, a SIP application can set the RequestURI to:

- sip:1234567890@ocsb-psx.net:5060;noa=national
- sip:+972540987610@ocsb-psx.net:5060;

SIP NOTIFY Message Body Formats

Service Broker uses the SIP NOTIFY message body to pass a MAP result to the SIP application. Service Broker passes the MAP result in one of the following formats:

- "application/map-phase3+xml"

The SIP NOTIFY message body contains a full MAP operation result, encoded in XER.
- "application/map-phase3+ber"

The SIP NOTIFY message body contains a full MAP operation result, encoded in BER.

- "application/pidf+xml"
The SIP NOTIFY message body contains the mobile subscriber's state and location in PIDF format.

Specifying Supported SIP NOTIFY Message Body Formats

SIP applications use the Accept header to specify the SIP NOTIFY message body formats that the application supports. The value of the Accept header must be one that is also supported by the Service Broker, that is values configured under the IM-PSX AcceptHeadersMBean.

For example, "application/map-phase3+xml" and "application/pidf+xml". See "[SIP NOTIFY Message Body Formats](#)" for more information.

The Accept header is optional. If not specified, Service Broker assumes that the SIP application supports the body formats specified in the IM-PSX AcceptHeadersMBean.

Setting the Expires Header

SIP applications must always set the Expires header to zero. If an application does not set the Expires header, Service Broker assumes this header is set to zero.

Handling SIP Errors

Service Broker can return all standard SIP errors. [Table 7-2](#) provides additional interpretation of some standard SIP errors specifically for Presence and Subscriber eXtensions applications.

Table 7-2 SIP Errors

Error	Description
400 Bad Request	The application sets the value of the Expires header to a value other than zero.
403 Forbidden	The application sets the value of the domain part in the requestURI which is different from the PsxSipDomain.
415 Unsupported Media Type	The application sets the value of the Accept header to a format which Service Broker does not support.
489 Bad Event	This error may indicate one of the following problems: <ul style="list-style-type: none"> ▪ The application sets the Event header with an unsupported value ▪ The application does not set the Event header at all ▪ The application sets the requested-info token with an unsupported value (see "Event Header")
500 Server Internal Error	Unexpected internal error has occurred in Service Broker.
603 Declined	There's a mismatch between the Accept header value and the requested-info token value. For example, if the application requests mobile subscriber's location (sets the requested-info token to 'Mobile-location'), but does not support XER format (does not set "application/map-phase3+xml" in the Accept header) that is required to receive the mobile subscriber's location.

Obtaining Subscriber's State and Location

Using the NG-IN solution, SIP applications can obtain a mobile subscriber's state and location that is stored in a network's HLR. Such information includes:

- Subscriber's state. For example, reachable or busy.
- Subscriber's location. For example, geographical location and VLR number.
- Other subscriber's information. For example, extensions information.

The ability to obtain a mobile subscriber's state and location information is based on and the MAP ANY-TIME-INTERROGATION operation. SIP applications can request Service Broker to obtain a mobile subscriber's state and location information by using the SIP SUBSCRIBE message. Service Broker returns the state and location information inside the SIP NOTIFY message body.

The following sections describe how a SIP application needs to generate a SIP SUBSCRIBE message to Service Broker in order to obtain a subscriber's state and location, and then process the SIP NOTIFY message which Service Broker sends back to the SIP application.

Generating a SIP SUBSCRIBE Message

The following SIP headers must be set in the SIP SUBSCRIBE message:

Common SIP Headers

- RequestURI
For more information, see ["Specifying the Identity of a Mobile Subscriber"](#)
- To
For more information, see ["Specifying the Address of an SS7 Entity"](#)
- Accept
For more information, see ["SIP NOTIFY Message Body Formats"](#)
- Expires
For more information, see ["Setting the Expires Header"](#)

Event Header

The SIP application must set the Event header to 'Presence'. In addition, the SIP application may define the **requested-info** token to specify the information that Service Broker needs to request from an HLR. For example, the application may request information only about the subscriber state without information about its location.

The SIP application can set **requested-info** token to one of the following values:

- Mobile-state
- Mobile-location

If the SIP application does not specify the **requested-info** token, Service Broker returns both the mobile subscriber's state and location.

Processing a SIP NOTIFY Request

Service Broker returns a mobile subscriber's state and location information inside a SIP NOTIFY message body. Depending on the information that the SIP application has

requested from Service Broker, the SIP NOTIFY message body may contain either one of the following or both:

- Subscriber's state - provided in PIDF format
- Subscriber's location and any other information - provided in the XER or BER format

This section describes important SIP headers and the SIP message body that a SIP application receives from Service Broker.

Subscription-State Header

The Subscription-State header value is always 'terminated'.

To explain the reason why a subscriber's state and location cannot be returned, the SIP application uses the **reason** token. Table 7-3 lists the possible values of the **reason** token:

Table 7-3 Possible Values of the Reason Token

Token Value	Description
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
map_unknowndsubscriber	HLR does not recognize the subscriber whose subscription information has been requested.
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_systemfailure	There is a problem to connect the HLR.
unknown	Unexpected internal Service Broker error occurred.
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_atinotallowed	HLR does not permit interrogation using the MAP-ANY-TIME-INTERROGATION operation.
map_timeout	HLR does not respond.

Content-Type Header

The Content-Type header specifies the format of the SIP NOTIFY message body, as follows:

- "application/map-phase3+xml" - the body contains the full MAP-ANY-TIME-INTERROGATION operation result structure encoded in the XER format.
- "application/map-phase3+ber" - the body contains the full MAP-ANY-TIME-INTERROGATION operation result encoded in the BER format.
- "application/pidf+xml" - the body contains subscriber's state encoded in PIDF format.
- "multipart/mixed; boundary='frontier'" - the body contains multiple formats, both the subscriber's state encoded in PIDF and the full MAP-ANY-TIME-INTERROGATION operation result structure encoded in XER.

SIP Message Body

The SIP NOTIFY message body contains information about subscriber's state and location as they were requested by the SIP application in the Event header and **requested-info** token of the SIP SUBSCRIBE message (for more information, see "Event Header").

The information requested by the SIP application is delivered in the message body as follows:

- Subscriber's state is provided in an XML according to the PIDF schema. For more information, see "Subscriber's State".
- Subscriber's location and additional subscriber information is provided in the XER or BER format. For more information, see "Other Subscriber Information".

Subscriber's State

Subscriber's state is provided in an XML, according to the PIDF schema.

The following example shows how information about the subscriber's state is encoded:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf "
xmlns:ts="urn:ietf:params:xml:ns:pidf:terminal-status"
entity="pres:123456789@psx-ocsb.net">
  <tuple id="sg89ae">
    <status>
      <basic>open</basic>
      <ts:state>reachable</ts:state>
    </status>
    <timestamp>2008-04-01T18:08:20Z</timestamp>
  </tuple>
</presence>
```

Table 7-4 explains the PIDF elements and attributes.

Table 7-4 PIDF Elements

Element	Description
<i>presence</i>	The root element. The element includes the <i>entity</i> attribute and 0 or more <i>tuple</i> elements. The value of the entity attribute contains the value of the RequestURI received on the SIP SUBSCRIBE message with 'pres' uri-scheme.
<i>tuple</i>	Contains the mobile subscriber's state information that consists of a mandatory <i>status</i> element accompanied by the optional <i>timestamp</i> element.
<i>status</i>	Contains one optional <i>basic</i> element accompanied by the <i>state</i> element.
<i>basic</i>	Specifies a subscriber's availability for communications. Possible values: <ul style="list-style-type: none"> ■ Open ■ Close

Table 7-4 (Cont.) PIDF Elements

Element	Description
<i>ts:state</i>	<p>Extension element that specifies the subscriber's state.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ Reachable ■ Unreachable ■ Busy ■ Unknown <p>The value of <i>ts:state</i> correlates with the value of <i>basic</i> as follows:</p> <ul style="list-style-type: none"> ■ When <i>basic</i> contains "Open", <i>ts:state</i> contains "Reachable". ■ When <i>basic</i> contains "Close", <i>ts:state</i> may contain "Unreachable", "Busy", or "Unknown".
<i>timestamp</i>	Specifies the date and time when the presence information was created.

Other Subscriber Information

Subscriber's location and other information is provided in the XER or BER format. In this case, the SIP NOTIFY message body contains the full MAP operation result structure.

Obtaining Mobile Subscriber's Subscription Information

Using the NG-IN solution, SIP applications can obtain mobile subscriber's subscription information that is stored in an HLR. Such information includes:

- Subscriber's basic information, for example, IMSI
- Subscriber's service information, for example, indication on services that are invoked for incoming and outgoing calls, mobility changes, incoming and outgoing SMS.

The ability to modify this information is based on the following MAP operations:

- MAP-ANY-TIME-SUBSCRIPTION-INTERROGATION
- MAP-SEND-IMSI

The ability to obtain subscription information is based on MAP operations. To obtain subscription information, a SIP application has to construct a XER or BER representation of the MAP operation request, and pass it to Service Broker inside the SIP SUBSCRIBE message body. Service Broker returns the subscription information in the SIP NOTIFY message body.

The following sections specify SIP interface requirements for subscription information interrogation, in addition to the common requirements specified at "[Understanding Common SIP Interface Concepts](#)".

Generating a SIP SUBSCRIBE Message

The following SIP headers must be set in the SIP SUBSCRIBE message:

Common SIP Headers

- RequestURI

For more information, see "[Specifying the Identity of a Mobile Subscriber](#)".

- To
For more information, see ["Specifying the Address of an SS7 Entity"](#)
- Accept
For more information, see ["SIP NOTIFY Message Body Formats"](#)
- Expires
For more information, see ["Setting the Expires Header"](#)

Event Header

SIP applications must set the Event header to 'SubQuery'.

Content-Type Header

SIP applications use the Content-Type header to specify the SIP SUBSCRIBE message body format, that is the MAP operation encoding format. Possible values are:

- "application/map-phase3+xml" - when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in XER
- "application/map-phase3+ber" - when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in BER

The SIP application should use two additional tokens to provide an indication of the MAP operation encoded inside the SIP message body:

- **op**, which defines the MAP operation code. You can set this token to one of the following values depending on the operation you want to trigger:
 - 58, when you want to trigger MAP-SEND-IMSI
 - 62, when you want to trigger
MAP-ANY-TIME-SUBSCRIPTION-INTERROGATION
- **dir**, which defines the MAP operation direction. Set this token to "invoke".

For example:

Content-Type: "application/map-phase3+xml; op=62; dir=invoke"

Processing the SIP NOTIFY Message

Service Broker returns a mobile subscriber's subscription information inside a SIP NOTIFY message body. Service Broker passes the MAP operation result, as was received from the HLR, encoded in XER or BER. The SIP application can request a preferable format, using the Accept header. See ["Specifying Supported SIP NOTIFY Message Body Formats"](#) for more information.

The following SIP headers can provide additional information to the MAP operation result in the SIP message body.

Subscription-State Header

The Subscription-State header value is always 'terminated'.

In case of a problem, the SIP application can use the **reason** token to identify the failure reason. [Table 7-5](#) lists the possible **reason** token values:

Table 7-5 Possible Values of the Reason Token

Token Value	Description
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
map_unknownsubscriber	HLR does not recognize the subscriber whose subscription information has been requested.
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_systemfailure	There is a problem to connect the HLR.
unknown	Unexpected internal Service Broker error occurred. An application may also receive this error if the XER or BER provided in the SIP SUBSCRIBE message body cannot be decoded to a MAP message.
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_atsinotallowed	HLR does not permit interrogation using the MAP-ANY-TIME-SUBSCRIPTION-INTERROGATION operation.
map_bearerservicenotprovisioned	The bearer service for which information is requested was not provisioned in the HLR.
map_teleservicenotprovisioned	The teleservice for which information is requested, was not provisioned.
map_illegalssoperation	Illegal supplementary service operation.
map_ssnotavailable	Supplementary service is not available.
map_informationnotavailable	The requested information is not available.

Modifying Mobile Subscriber's Information

Using the NG-IN solution, SIP applications can modify subscriber's data in an HLR or VLR. The ability to modify this information is based on the following MAP operations:

- MAP-ANY-TIME-MODIFICATION
- MAP-INSERT-SUBSCRIBER-DATA

SIP applications can request Service Broker to modify subscription information using the SIP SUBSCRIBE message. Service Broker returns the result of the modification operation inside the SIP NOTIFY message body.

The following sections describe how a SIP application needs to generate a SIP SUBSCRIBE message to Service Broker in order to modify subscriber's data, and then process the SIP NOTIFY message which Service Broker sends back to the SIP application.

Generating a SIP Subscribe Message

The following sections describe the SIP headers that must be set in the SIP SUBSCRIBE message.

Common SIP Headers

- RequestURI
For more information, see ["Specifying the Identity of a Mobile Subscriber"](#).
- To
For more information, see ["Specifying the Address of an SS7 Entity"](#).
- Accept
For more information, see ["SIP NOTIFY Message Body Formats"](#).
- Expires
For more information, see ["Setting the Expires Header"](#).

Event Header

The SIP application must set the Event header to 'SubUpdate'.

Content-Type Header

SIP applications use the Content-Type header to specify the SIP SUBSCRIBE message body format, that is the MAP operation encoding format. The header can contain one of the following values:

- "application/map-phase3+xml", when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in XER
- "application/map-phase3+ber", when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in BER

The SIP application can use two additional tokens to provide an indication of the MAP operation encoded inside the SIP message body:

- **op**, which defines the MAP operation code. You can set this token to one of the following values depending on the operation you want to trigger:
 - 7, when you want to trigger MAP-INSERT-SUBSCRIBER-DATA
 - 65, when you want to trigger MAP-ANY-TIME-MODIFICATION
- **dir**, which defines the MAP operation direction. Set this token to "invoke".

For example:

Content-Type: "application/map-phase3+xml; op=65; dir=invoke"

If the body is empty, Service Broker returns the SIP error 400 'Bad request'.

Processing a SIP Notify Message

Service Broker returns a result of a modification operation inside a SIP NOTIFY message body. This section describes the SIP headers and the SIP message body that a SIP application receives from Service Broker.

Subscription-State Header

The Subscription-State header value is always 'terminated'. If a problem occurs, the SIP application can set the **reason** token to the values described in [Table 7-6](#).

Table 7-6 Possible Values of the Reason Token

Token Value	Description
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_bearerservicenotprovisioned	The bearer service for which information is requested was not provisioned in the HLR.
map_callbarred	Operator determined barring or supplementary service barring management is active
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_illegalsoperation	Illegal supplementary service operation.
map_informationnotavailable	The requested information is not available.
map_serrorstatus	Supplementary service error status
map_ssincompatibility	Supplementary service incompatibility
map_sssubscriptionviloation	Supplementary service subscription violation
map_teleservicenotprovisioned	The teleservice for which information is requested, was not provisioned.
map_atmnotallowed	MAP ANY-TIME-MODIFICATION operation is not allowed by the HLR
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_unknownssubscriber	HLR does not recognize the subscriber whose subscription information was requested.
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
unknown	Unexpected internal Service Broker error occurred. An application may also receive this error if the XER or BER provided in the SIP SUBSCRIBE message body cannot be decoded to a MAP message.

Content-Type Header

The Content-Type header specifies the format of the SIP NOTIFY message body, as follows:

- "application/map-phase3+xml", when the body contains the full modification operation result structure encoded in the XER format
- "application/map-phase3+ber", when the body contains the full modification operation result structure encoded in the BER format

Developing a Presence and Subscriber Status SIP Application

This chapter describes how to develop a Presence and Subscriber Status eXtensions SIP application in the Oracle Communications Service Broker NG-IN solution.

Understanding Common SIP Interface Concepts

Using the NG-IN solution, SIP applications can obtain mobile subscriber's information that is stored in SS7 network entities such as HLR. To obtain subscriber's information, a SIP application has trigger a SIP SUBSCRIBE message to Service Broker and specify inside the information that it needs. Service Broker returns the subscriber's information in a SIP NOTIFY message.

The following sections describe common parts of the SIP SUBSCRIBE and NOTIFY interface, such as headers and SIP errors, that the SIP application generates every time it communicates with Service Broker, regardless of the type of operation the SIP application requests to perform.

Specifying the Address of an SS7 Entity

SIP applications can instruct Service Broker which SS7 entity to connect using the domain part of the To header. An application can set the domain part using one of the following methods:

- Setting an alias

When a SIP application needs to communicate with an SS7 entity whose SCCP address is configured in the SS7 SSU, the application sets an alias that refers to this address.

Service Broker uses this alias to resolve the preconfigured SCCP address (that is point code or GT address). For example, if you set the To header to **sip:1234567890@h1r01**, then Service Broker will resolve h1r01 to a real SCCP address, based on the SCCP addresses configured in the SS7 SSU.

Note that if you specify an alias that resolves to a dynamic GT address, then you must also specify the GT digits, using the 'gtaddr' token. For example, **sip:1234567890@h1r01;gtaddr=972543349098**

- Using an IM-PSX SIP domain

When a SIP application needs to communicate with an SS7 entity whose alias is preconfigured in IM-PSX, in the **PsxSipDomain** parameter, the application sets the IM-PSX SIP domain.

When you set the IM-PSX SIP domain in the domain part of the To header, for example, **sip:1234567890@ocsb-psx.net**, Service Broker uses the alias configured in the **DefaultSs7EntityAlias** parameter to resolve an SCCP address that is already configured in the SS7 SSU.

- Using the Anonymous string

If a SIP application needs to communicate with an entity whose SS7 address is not configured in Service Broker, the application constructs the domain part of the To header using the "Anonymous" string.

Setting the "Anonymous" string means the application uses additional tokens in the To header to specify an SCCP address.

The "Anonymous" string requires an application to set the tokens described in [Table 8-1](#).

Table 8-1 Tokens Required to Specify an SCCP Address

Token	Type	Description
gtaddr	STRING	Stands for Global Title Address. The parameter contains digits.
nai	STRING	Stands for Nature of Address Indicator. Possible values: <ul style="list-style-type: none"> ■ unknown ■ subscriberNumber ■ nationalReserved ■ nationalSignificant ■ international
np	STRING	Stands for Numbering Plan. Possible values: <ul style="list-style-type: none"> ■ unknown ■ isdn ■ generic ■ data ■ telex ■ maritimeMobile ■ landMobile ■ isdnMobile
netind	STRING	Stands for Network Indicator. Possible values: <ul style="list-style-type: none"> ■ national ■ international
tt	BYTE	Stands for Translation Type.
spc	Up to 24 bit decimal	Stands for Signaling Point Code.
ssn	BYTE	Stands for Subsystem Number.

An application must set the tokens described in [Table 8-1](#) as follows:

- The **netind** token is always required.
- One of the following tokens is required:
 - **gtaddr**. If an application uses the **gtaddr** token, to allow Service Broker to build the global title indicator, the application must set at least one of the following tokens: **tt** or **nai**.
 - **spc** accompanied by the **ssn** token

For example, the following combinations of tokens are considered valid:

- **netind, gtaddr, nai, np, tt**
- **netind, gtaddr, tt**
- **netind, gtaddr, tt, np**
- **netind, gtaddr, nai**
- **netind, spc, ssn**
- **netind, gtaddr, nai, np, tt, spc, ssn**

Specifying the Identity of a Mobile Subscriber

SIP applications specify the mobile subscriber whose information is required by using the domain part of the RequestURI.

SIP applications should set the RequestURI as follows:

- Set the mobile subscriber's MSISDN in the user part
- Set the IM-PSX address, as configured in **PsxSipDomain**, in the domain part.

A SIP application can use the **noa** token, to specify the MSISDN nature of address that is later used on the MAP interface. The possible **noa** token values are:

- subscriber
- unknown
- national

To specify that the MSISDN nature of address is 'international', the SIP application must set the MSISDN in the RequestURI user part in a global format, with a leading '+' sign.

For example, a SIP application can set the RequestURI to:

- sip:1234567890@ocsb-psx.net:5060;noa=national
- sip:+972540987610@ocsb-psx.net:5060;

SIP NOTIFY Message Body Formats

Service Broker uses the SIP NOTIFY message body to pass a MAP result to the SIP application. Service Broker passes the MAP result in one of the following formats:

- "application/map-phase3+xml"

The SIP NOTIFY message body contains a full MAP operation result, encoded in XER.
- "application/map-phase3+ber"

The SIP NOTIFY message body contains a full MAP operation result, encoded in BER.

- "application/pidf+xml"
The SIP NOTIFY message body contains the mobile subscriber's state and location in PIDF format.

Specifying Supported SIP NOTIFY Message Body Formats

SIP applications use the Accept header to specify the SIP NOTIFY message body formats that the application supports. The value of the Accept header must be one that is also supported by the Service Broker, that is values configured under the IM-PSX AcceptHeadersMBean.

For example, "application/map-phase3+xml" and "application/pidf+xml". See "[SIP NOTIFY Message Body Formats](#)" for more information.

The Accept header is optional. If not specified, Service Broker assumes that the SIP application supports the body formats specified in the IM-PSX AcceptHeadersMBean.

Setting the Expires Header

SIP applications must always set the Expires header to zero. If an application does not set the Expires header, Service Broker assumes this header is set to zero.

Handling SIP Errors

Service Broker can return all standard SIP errors. [Table 8-2](#) provides additional interpretation of some standard SIP errors specifically for Presence and Subscriber eXtensions applications.

Table 8-2 SIP Errors

Error	Description
400 Bad Request	The application sets the value of the Expires header to a value other than zero.
403 Forbidden	The application sets the value of the domain part in the requestURI which is different from the PsxSipDomain.
415 Unsupported Media Type	The application sets the value of the Accept header to a format which Service Broker does not support.
489 Bad Event	This error may indicate one of the following problems: <ul style="list-style-type: none"> ▪ The application sets the Event header with an unsupported value ▪ The application does not set the Event header at all ▪ The application sets the requested-info token with an unsupported value.
500 Server Internal Error	Unexpected internal error has occurred in Service Broker.
603 Declined	There's a mismatch between the Accept header value and the requested-info token value. For example, if the application requests mobile subscriber's location (sets the requested-info token to 'Mobile-location'), but does not support XER format (does not set "application/map-phase3+xml" in the Accept header) that is required to receive the mobile subscriber's location.

Obtaining Subscriber's State and Location

Using the NG-IN solution, SIP applications can obtain a mobile subscriber's state and location that is stored in a network's HLR. Such information includes:

- Subscriber's state. For example, reachable or busy.
- Subscriber's location. For example, geographical location and VLR number.
- Other subscriber's information. For example, extensions information.

The ability to obtain a mobile subscriber's state and location information is based on and the MAP ANY-TIME-INTERROGATION operation. SIP applications can request Service Broker to obtain a mobile subscriber's state and location information by using the SIP SUBSCRIBE message. Service Broker returns the state and location information inside the SIP NOTIFY message body.

The following sections describe how a SIP application needs to generate a SIP SUBSCRIBE message to Service Broker in order to obtain a subscriber's state and location, and then process the SIP NOTIFY message which Service Broker sends back to the SIP application.

Generating a SIP SUBSCRIBE Message

The following SIP headers must be set in the SIP SUBSCRIBE message:

Common SIP Headers

- RequestURI
For more information, see ["Specifying the Identity of a Mobile Subscriber"](#)
- To
For more information, see ["Specifying the Address of an SS7 Entity"](#)
- Accept
For more information, see ["SIP NOTIFY Message Body Formats"](#)
- Expires
For more information, see ["Setting the Expires Header"](#)

Event Header

The SIP application must set the Event header to 'Presence'. In addition, the SIP application may define the **requested-info** token to specify the information that Service Broker needs to request from an HLR. For example, the application may request information only about the subscriber state without information about its location.

The SIP application can set **requested-info** token to one of the following values:

- Mobile-state
- Mobile-location

If the SIP application does not specify the **requested-info** token, Service Broker returns both the mobile subscriber's state and location.

Processing a SIP NOTIFY Request

Service Broker returns a mobile subscriber's state and location information inside a SIP NOTIFY message body. Depending on the information that the SIP application has requested from Service Broker, the SIP NOTIFY message body may contain either one of the following or both:

- Subscriber's state - provided in PIDF format
- Subscriber's location and any other information - provided in the XER or BER format

This section describes important SIP headers and the SIP message body that a SIP application receives from Service Broker.

Subscription-State Header

The Subscription-State header value is always 'terminated'.

To explain the reason why a subscriber's state and location cannot be returned, the SIP application uses the **reason** token. [Table 8-3](#) lists the possible values of the **reason** token:

Table 8-3 Possible Values of the Reason Token

Token Value	Description
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
map_unknownsubscriber	HLR does not recognize the subscriber whose subscription information has been requested.
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_systemfailure	There is a problem to connect the HLR.
unknown	Unexpected internal Service Broker error occurred.
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_atinotallowed	HLR does not permit interrogation using the MAP-ANY-TIME-INTERROGATION operation.
map_timeout	HLR does not respond.

Content-Type Header

The Content-Type header specifies the format of the SIP NOTIFY message body, as follows:

- "application/map-phase3+xml" - the body contains the full MAP-ANY-TIME-INTERROGATION operation result structure encoded in the XER format.
- "application/map-phase3+ber" - the body contains the full MAP-ANY-TIME-INTERROGATION operation result encoded in the BER format.
- "application/pidf+xml" - the body contains subscriber's state encoded in PIDF format.

- "multipart/mixed; boundary='frontier'" - the body contains multiple formats, both the subscriber's state encoded in PIDF and the full MAP-ANY-TIME-INTERROGATION operation result structure encoded in XER.

SIP Message Body

The SIP NOTIFY message body contains information about subscriber's state and location as they were requested by the SIP application in the Event header and **requested-info** token of the SIP SUBSCRIBE message (for more information, see "Event Header").

The information requested by the SIP application is delivered in the message body as follows:

- Subscriber's state is provided in an XML according to the PIDF schema. For more information, see "Subscriber's State".
- Subscriber's location and additional subscriber information is provided in the XER or BER format. For more information, see "Other Subscriber Information".

Subscriber's State

Subscriber's state is provided in an XML, according to the PIDF schema.

The following example shows how information about the subscriber's state is encoded:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf "
xmlns:ts="urn:ietf:params:xml:ns:pidf:terminal-status"
entity="pres:123456789@psx-ocsb.net">
  <tuple id="sg89ae">
    <status>
      <basic>open</basic>
      <ts:state>reachable</ts:state>
    </status>
    <timestamp>2008-04-01T18:08:20Z</timestamp>
  </tuple>
</presence>
```

Table 8–4 explains the PIDF elements and attributes.

Table 8–4 PIDF Elements

Element	Description
<presence>	The root element. The element includes the <entity> attribute and 0 or more <tuple> elements. The value of the entity attribute contains the value of the RequestURI received on the SIP SUBSCRIBE message with 'pres' uri-scheme.
<tuple>	Contains the mobile subscriber's state information that consists of a mandatory <status> element accompanied by the optional <timestamp> element.
<status>	Contains one optional <basic> element accompanied by the <state> element.
<basic>	Specifies a subscriber's availability for communications. Possible values: <ul style="list-style-type: none"> ■ Open ■ Close

Table 8–4 (Cont.) PIDF Elements

Element	Description
<ts:state>	<p>Extension element that specifies the subscriber's state.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ Reachable ■ Unreachable ■ Busy ■ Unknown <p>The value of <ts:state> correlates with the value of <basic> as follows:</p> <ul style="list-style-type: none"> ■ When <basic> contains "Open", <ts:state> contains "Reachable". ■ When <basic> contains "Close", <ts:state> may contain "Unreachable", "Busy", or "Unknown".
<timestamp>	Specifies the date and time when the presence information was created.

Other Subscriber Information

Subscriber's location and other information is provided in the XER or BER format. In this case, the SIP NOTIFY message body contains the full MAP operation result structure.

Obtaining Mobile Subscriber's Subscription Information

Using the NG-IN solution, SIP applications can obtain mobile subscriber's subscription information that is stored in an HLR. Such information includes:

- Subscriber's basic information, for example, IMSI
- Subscriber's service information, for example, indication on services that are invoked for incoming and outgoing calls, mobility changes, incoming and outgoing SMS.

The ability to modify this information is based on the following MAP operations:

- ANY-TIME-SUBSCRIPTION-INTERROGATION
- MAP-SEND-IMSI

The ability to obtain subscription information is based on MAP operations. To obtain subscription information, a SIP application has to construct a XER or BER representation of the MAP operation request, and pass it to Service Broker inside the SIP SUBSCRIBE message body. Service Broker returns the subscription information in the SIP NOTIFY message body.

The following sections specify SIP interface requirements for subscription information interrogation, in addition to the common requirements specified at "[Understanding Common SIP Interface Concepts](#)".

Generating a SIP SUBSCRIBE Message

The following SIP headers must be set in the SIP SUBSCRIBE message:

Common SIP Headers

- RequestURI

For more information, see ["Specifying the Identity of a Mobile Subscriber"](#)

- To

For more information, see ["Specifying the Address of an SS7 Entity"](#)

- Accept

For more information, see ["SIP NOTIFY Message Body Formats"](#)

- Expires

For more information, see ["Setting the Expires Header"](#)

Event Header

SIP applications must set the Event header to 'SubQuery'.

Content-Type Header

SIP applications use the Content-Type header to specify the SIP SUBSCRIBE message body format, that is the MAP operation encoding format. Possible values are:

- "application/map-phase3+xml" - when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in XER
- "application/map-phase3+ber" - when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in BER

The SIP application should use two additional tokens to provide an indication of the MAP operation encoded inside the SIP message body:

- **op**, which defines the MAP operation code. You can set this token to one of the following values depending on the operation you want to trigger:
 - 58, when you want to trigger MAP-SEND-IMSI
 - 62, when you want to trigger MAP-ANY-TIME-SUBSCRIPTION-INTERROGATION
- **dir**, which defines the MAP operation direction. Set this token to "invoke".

For example:

Content-Type: "application/map-phase3+xml; op=62; dir=invoke"

Processing the SIP NOTIFY Message

Service Broker returns a mobile subscriber's subscription information inside a SIP NOTIFY message body. Service Broker passes the MAP operation result, as was received from the HLR, encoded in XER or BER. The SIP application can request a preferable format, using the Accept header. See ["Specifying Supported SIP NOTIFY Message Body Formats"](#) for more information.

The following SIP headers can provide additional information to the MAP operation result in the SIP message body.

Subscription-State Header

The Subscription-State header value is always 'terminated'.

In case of a problem, the SIP application can use the **reason** token to identify the failure reason. [Table 8-5](#) lists the possible **reason** token values:

Table 8–5 Possible Values of the Reason Token

Token Value	Description
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
map_unknownsubscriber	HLR does not recognize the subscriber whose subscription information has been requested.
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_systemfailure	There is a problem to connect the HLR.
unknown	Unexpected internal Service Broker error occurred. An application may also receive this error if the XERorBER provided in the SIP SUBSCRIBE message body cannot be decoded to a MAP message.
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_atsinotallowed	HLR does not permit interrogation using the MAP-ANY-TIME-SUBSCRIPTION-INTERROGATION operation.
map_bearerservicenotprovisioned	The bearer service for which information is requested was not provisioned in the HLR.
map_teleservicenotprovisioned	The teleservice for which information is requested, was not provisioned.
map_illegalssoperation	Illegal supplementary service operation.
map_ssnotavailable	Supplementary service is not available.
map_informationnotavailable	The requested information is not available.

Modifying Mobile Subscriber's Information

Using the NG-IN solution, SIP applications can modify subscriber's data in an HLR or VLR. The ability to modify this information is based on the following MAP operations:

- MAP-ANY-TIME-MODIFICATION
- MAP-INSERT-SUBSCRIBER-DATA

SIP applications can request Service Broker to modify subscription information using the SIP SUBSCRIBE message. Service Broker returns the result of the modification operation inside the SIP NOTIFY message body.

The following sections describe how a SIP application needs to generate a SIP SUBSCRIBE message to Service Broker in order to modify subscriber's data, and then process the SIP NOTIFY message which Service Broker sends back to the SIP application.

Generating a SIP Subscribe Message

The following sections describe the SIP headers that must be set in the SIP SUBSCRIBE message.

Common SIP Headers

- RequestURI
For more information, see ["Specifying the Identity of a Mobile Subscriber"](#).
- To
For more information, see ["Specifying the Address of an SS7 Entity"](#).
- Accept
For more information, see ["SIP NOTIFY Message Body Formats"](#).
- Expires
For more information, see ["Setting the Expires Header"](#).

Event Header

The SIP application must set the Event header to 'SubUpdate'.

Content-Type Header

SIP applications use the Content-Type header to specify the SIP SUBSCRIBE message body format, that is the MAP operation encoding format. The header can contain one of the following values:

- "application/map-phase3+xml", when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in XER
- "application/map-phase3+ber", when the MAP operation request inside the SIP SUBSCRIBE message body is encoded in BER

The SIP application can use two additional tokens to provide an indication of the MAP operation encoded inside the SIP message body:

- **op**, which defines the MAP operation code. You can set this token to one of the following values depending on the operation you want to trigger:
 - 7, when you want to trigger MAP-INSERT-SUBSCRIBER-DATA
 - 65, when you want to trigger MAP-ANY-TIME-MODIFICATION
- **dir**, which defines the MAP operation direction. Set this token to "invoke".

For example:

Content-Type: "application/map-phase3+xml; op=65; dir=invoke"

If the body is empty, Service Broker returns the SIP error 400 'Bad request'.

Processing a SIP Notify Message

Service Broker returns a result of a modification operation inside a SIP NOTIFY message body. This section describes the SIP headers and the SIP message body that a SIP application receives from Service Broker.

Subscription-State Header

The Subscription-State header value is always 'terminated'. If a problem occurs, the SIP application can set the **reason** token to the values described in [Table 8-6](#).

Table 8–6 Possible Values of the Reason Token

Token Value	Description
addressresolutionfailure	Service Broker failed to resolve the SCCP address alias, that is the SS7 entity address.
map_bearerservicenotprovisioned	The bearer service for which information is requested was not provisioned in the HLR.
map_callbarred	Operator determined barring or supplementary service barring management is active
map_datamissing	HLR stated that the some data is missing in the MAP operation request.
map_illegalsoperation	Illegal supplementary service operation.
map_informationnotavailable	The requested information is not available.
map_serrorstatus	Supplementary service error status
map_ssincompatibility	Supplementary service incompatibility
map_sssubscriptionviloation	Supplementary service subscription violation
map_teleservicenotprovisioned	The teleservice for which information is requested, was not provisioned.
map_atmnotallowed	MAP ANY-TIME-MODIFICATION operation is not allowed by the HLR
map_unexpecteddatavalue	HLR found unexpected data in the MAP operation request.
map_unknownssubscriber	HLR does not recognize the subscriber whose subscription information was requested.
timeout	Interrogation terminated successfully. The result is available in the SIP NOTIFY message body.
unknown	Unexpected internal Service Broker error occurred. An application may also receive this error if the XER or BER provided in the SIP SUBSCRIBE message body cannot be decoded to a MAP message.

Content-Type Header

The Content-Type header specifies the format of the SIP NOTIFY message body, as follows:

- "application/map-phase3+xml", when the body contains the full modification operation result structure encoded in the XER format
- "application/map-phase3+ber", when the body contains the full modification operation result structure encoded in the BER format

Developing a Short Message Application

This chapter describes how to develop a SIP application that sends short messages to, and receives short messages from, mobile subscribers.

About Solution Architecture

Service Broker provides interfaces that allow you to develop a SIP application that sends short messages to, and receives messages from, mobile subscribers through a Short Message System Center (SMSC).

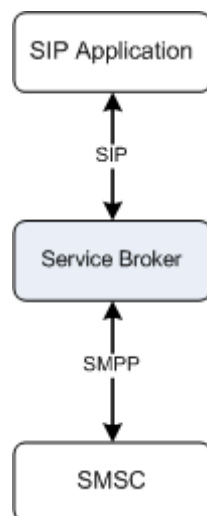
The application-facing side of Service Broker provides a SIP application with a standard SIP interface. The network-facing side of Service Broker provides an SMPP interface that enables Service Broker to communicate with SMSCs. Service Broker acts as a mediator by providing translation capabilities between SMPP and SIP.

A solution based on Service Broker consists of the following components:

- SIP application
- Service Broker
- Short Message System Center (SMSC)

Figure 9–1 shows a SIP application that interacts with an SMSC.

Figure 9–1 Architecture for Interacting with an SMSC



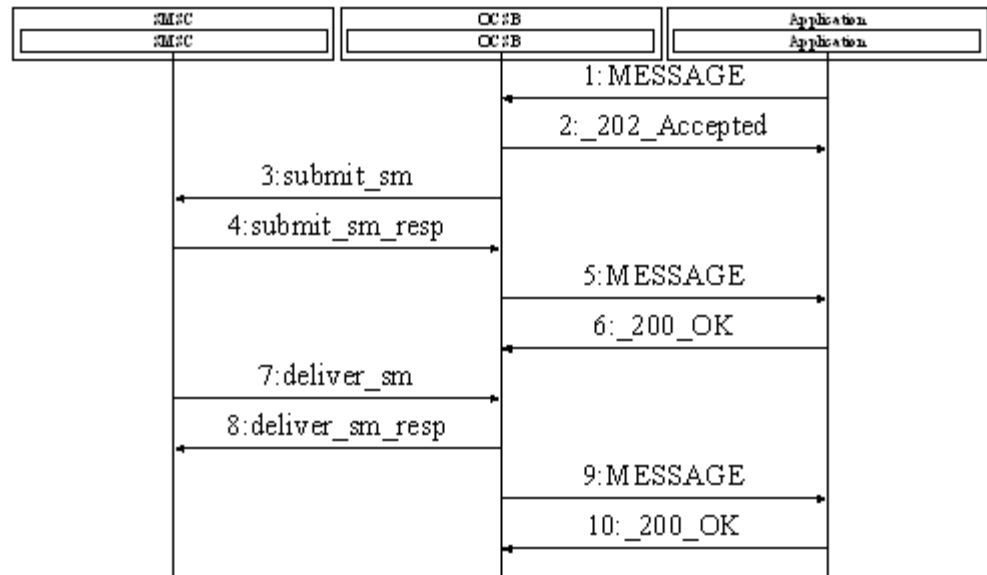
Sending a Short Message from an Application to an SMSC

When an application sends a message to an SMSC, the flow works as follows:

1. A SIP application sends a **SIP MESSAGE** to Service Broker.
2. Service Broker responds to the application with a **202 Accepted** message.
3. Based on the **SIP MESSAGE**, Service Broker generates a **submit_sm** message and sends it to the SMSC.
4. The SMSC responds to Service Broker with a **submit_sm_resp**.
5. Based on the **submit_sm_resp** message, Service Broker generates a **SIP MESSAGE** and sends it to the application.
6. The application responds to Service Broker with a **200 OK** message.
7. If Service Broker is configured to request a delivery receipt or acknowledgement from the SMSC, an SMSC sends a **deliver_sm** message to Service Broker. This message can carry one of the following:
 - SMSC Delivery receipt, which confirms that the SMSC received the message from the application.
 - SME Manual/User Acknowledgement, which is sent by a mobile device in response to the application's message. For example, this acknowledgement might contain a selected menu item from a menu list that the application sent to the mobile subscriber.
 - SME Delivery Acknowledgement, which confirms that the mobile subscriber has read the short message.
8. Service Broker responds to the SMSC with a **deliver_sm_resp** message.
9. Based on the **deliver_sm**, Service Broker sends a **SIP MESSAGE** to the application.
10. The application responds to Service Broker with a **200 OK** message.

Figure 9-2 shows an example flow when an application sends a short message to an SMSC.

Figure 9–2 Call Flow for Sending a Short Message to an SMSC



About Methods of Specifying Short Message Parameters

To translate a SIP MESSAGE to an appropriate SMPP message, Service Broker uses an IM-UIX-SMS interworking module. You can configure this module using the Service Broker Administration Console.

When the application encapsulates a short message into the **Content** header, IM-UIX-SMS generates an SMPP message using the information that you defined in headers of the SIP MESSAGE as well as in configuration settings of the IM-UIX-SMS itself.

Most of the parameters can be defined in both SIP MESSAGE headers and IM-UIX-SMS settings. If you do not provide parameters in the headers, Service Broker uses the settings defined in the configuration of the IM-UIX-SMS. When the settings are defined both in the headers and in the IM-UIX-SMS configuration, Service Broker uses the settings defined by the headers.

The following sections explain how to specify various parameters. In addition, each section explicitly specifies whether you can use headers or IM-UIX-SMS configuration settings to specify a parameter.

Specifying the Identity of an Application

You specify the identity of the application that sends a short message by using the user part of the **From** header. In addition, you might specify the optional tokens **Numbering Plan Indicator** (NPI) and **Nature of Address** (NOA).

Specifying the Identity of a Mobile Subscriber

You specify the identity of a mobile subscriber by specifying the MDN of the subscriber in the user part of the **RequestURI** header.

In addition, you can specify subscriber's NOA or NPI using one of the following methods:

- Specifying the **NOA** or **NPI** tokens of the **RequestURI** header

- Using the Service Broker Administration Console. You can define the following parameters on the IM-UIX-SMS configuration screen:
 - **Submit Destination Address Type of Number**
 - **Submit Destination Address Numbering Plan Identification**
- See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Specifying the Message Content Type

You can specify the content type using one of the following methods:

- Specifying the **Content-Type** header of a SIP MESSAGE. You must set **Content-Type** to "plain/text". Optionally, you can also set the **charset** parameter of the **Content-Type** header.
 - Using the Service Broker Administration Console. You need to define the **Data Coding** parameter on the IM-UIX-SMS configuration screen.
- See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Setting the Validity Period of a Message

The validity period of a message is the time period after which the message is discarded if it cannot be delivered to the destination.

Service Broker applies the validity period only when the **Submit Validity Period Format** is set to either **VP_RELATIVE_FORMAT** or **VP_ABSOLUTE_FORMAT**. You specify **Submit Validity Period Format** on the IM-UIX-SMS configuration screen.

You can specify the validity period using one of the following methods:

- Setting the **Expires** and **Date** headers. To make Service Broker to use the values set in these headers, on the IM-UIX-SMS configuration screen, you need to set the **Submit Validity Period Source** parameter to **SAL_MESSAGE**.
- Using the Service Broker Administration Console. Service Broker uses the values set in the IM-UIX-SMS configuration if the configuration parameters are set as follows:
 - In **Submit Validity Period Value**, specify the time after which the message is discarded.
 - Set **Submit Validity Period Source** to **STATIC_CONFIGURATION**.
 - Set **Submit Validity Period Format** to **VP_RELATIVE_FORMAT**.

See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Triggering an SMSC to Send a Predefined Short Message to a Mobile Subscriber

If an SMSC stores a collection of pre-defined short messages, you can trigger the SMSC to send one of these messages to a subscriber by using the **Play** header. In this header, you specify the ID of the message that you want the SMSC to send.

Specifying the Message Mode, Type, and Privacy Indicator

You can specify the message mode, type, and privacy indicator using one of the following methods:

- Setting the **Mode**, **Type**, and **Privacy** tokens of the **Subject** header
- Using the Service Broker Administration Console. You can define the following parameters on the IM-UIX-SMS configuration screen:
 - **Submit Message Mode**
 - **Submit Message Type**
 - **Privacy Indicator**

See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Specifying the Message Priority

You can specify the message priority using one of the following methods:

- Setting the **Priority** header
- Using the Service Broker Administration Console. You can define the **Submit SM Priority Level** parameter on the IM-UIX-SMS configuration screen.

See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Requesting a Delivery Receipt or Acknowledgement

You can configure Service Broker to request the SMSC to send a delivery receipt or acknowledgement. You can set these settings only by using the Service Broker Administration Console. You can define whether or not Service Broker should send a receipt or acknowledgement by specifying the following parameters on the IM-UIX-SMS configuration screen:

- **Delivery receipt request**
- **SME Acknowledgement Request**

See the "Configuring SMPP Operations" section of the "Configuring IM-UIX-SMS" chapter in *Oracle Communications Service Broker Processing Domain Configuration Guide* for more information.

Tunneling an SMPP Message using the XER or BER Format

An application can send a ready SMPP message to Service Broker. You need to encapsulate the message in the XER or BER format. In this case, Service Broker forwards the message to the SMSC.

You must set the Content-Type header to one of the following:

- **application/smpp-phase3.4+xml**, when you want to encapsulate a message in the XER format.
- **application/vnd.3gpp.sms**, when you want to encapsulate a message in the BER format.

Processing Headers of a Delivery Receipt or Acknowledgement

You can configure Service Broker to request the SMSC to send a delivery receipt or acknowledgement. After Service Broker received a receipt or acknowledgement from the SMSC, Service Broker generates a SIP MESSAGE and sends it to the application.

Table 9–1 describes headers and tokens that a SIP application needs to process in this message.

Table 9–1 Headers of a Delivery Receipt or Acknowledgement Received from an SMSC

Header	Description
In-Reply-To	Specifies the Call-ID of the original message. The application can use the In-Reply-To header to associate the response message with the original one.
Content	<p>Contains the state of the message for which Service Broker requested a delivery receipt.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ▪ 2 The message is delivered to the destination. ▪ 3 Validity period of the message expired. ▪ 4 Message was deleted. ▪ 5 Message is undeliverable. ▪ 6 The message has been manually read on behalf of the subscriber by a customer service. ▪ 7 The message is in an invalid state. ▪ 8 The message is in the rejected state.

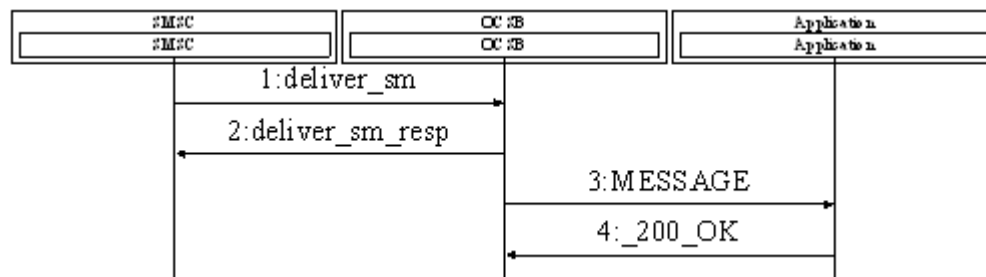
Receiving a Short Message Sent by a Mobile Subscriber

An SMSC initiates sending of a short message to an application when the SMSC needs to forwards a message sent by a mobile subscriber.

The flow of sending of a short message from an SMSC to an application works as follows:

1. An SMSC sends a **deliver_sm** message to Service Broker.
2. Based on the **deliver_sm**, Service Broker sends a **SIP MESSAGE** to the application.
3. The application responds to Service Broker with a **200 OK** message.

Figure 9–3 shows an example flow when an SMSC sends a short message to an application.

Figure 9–3 Call Flow for Sending a Short Message to an Application

Processing Headings of a Short Message Sent by a Mobile Subscriber

Table 9–1 describes headers and tokens that a SIP application needs to process in a SIP MESSAGE if the message is sent by a mobile subscriber.

Table 9–2 Headers of a Message Sent by a Mobile Subscriber

Header	Description
RequestURI, To	<ul style="list-style-type: none"> ▪ User part token: Specifies the address of the application to which the subscriber sends the message. ▪ Domain part token: Specifies the domain part of the application to which the subscriber sends the message. In addition, the RequestURI and To headers might contain the NOA and NPI tokens.
From	<ul style="list-style-type: none"> ▪ User part token: Specifies the MDN of the subscriber who sends the message. ▪ Domain part token: Specifies the domain part of the address of the subscriber who sends the message. ▪ NOA token: Specifies the Nature of Address of the address of the subscriber who sends the message. ▪ NPI token: Specifies the Numbering Plan Indicator of the address of the subscriber who sends the message.
Content-Type	Specifies the content type of the message.
Content-Language	Specifies the language of the response.
Content	If the Content-Type header is set to "plain/text", the Content header contains the contents of the short message.

Handling Errors

Service Broker sends a SIP MESSAGE to the application with the **Reason** header containing the cause of the error in the following cases:

- If sending of the original message failed, Service Broker sends a **SIP MESSAGE** to the application and adds the **Reason** header to this message. The header contains the error code. The Content header contains the description of the error.
- If Service Broker does not receive a delivery receipt or acknowledgement from the SMSC within the specified time period.

Detailed Sequence Diagram Examples

This appendix describes detailed sequence diagrams and message dumps of an application that provides a personalized ring back tone.

Personalized Ring Back Tone

The personalized ring back tone service is a terminating service that improves the traditional ring back tone by playing a personalized ring back tone to the calling party during the alerting phase of the call.

When the called party answers the call, the personalized ring back tone is stopped, and the parties are connected.

[Figure A-1](#), [Figure A-2](#) and [Figure A-3](#) show the detailed sequence diagram of the personalized ring back tone service.

Figure A-1 Personalized Ring Back Tone

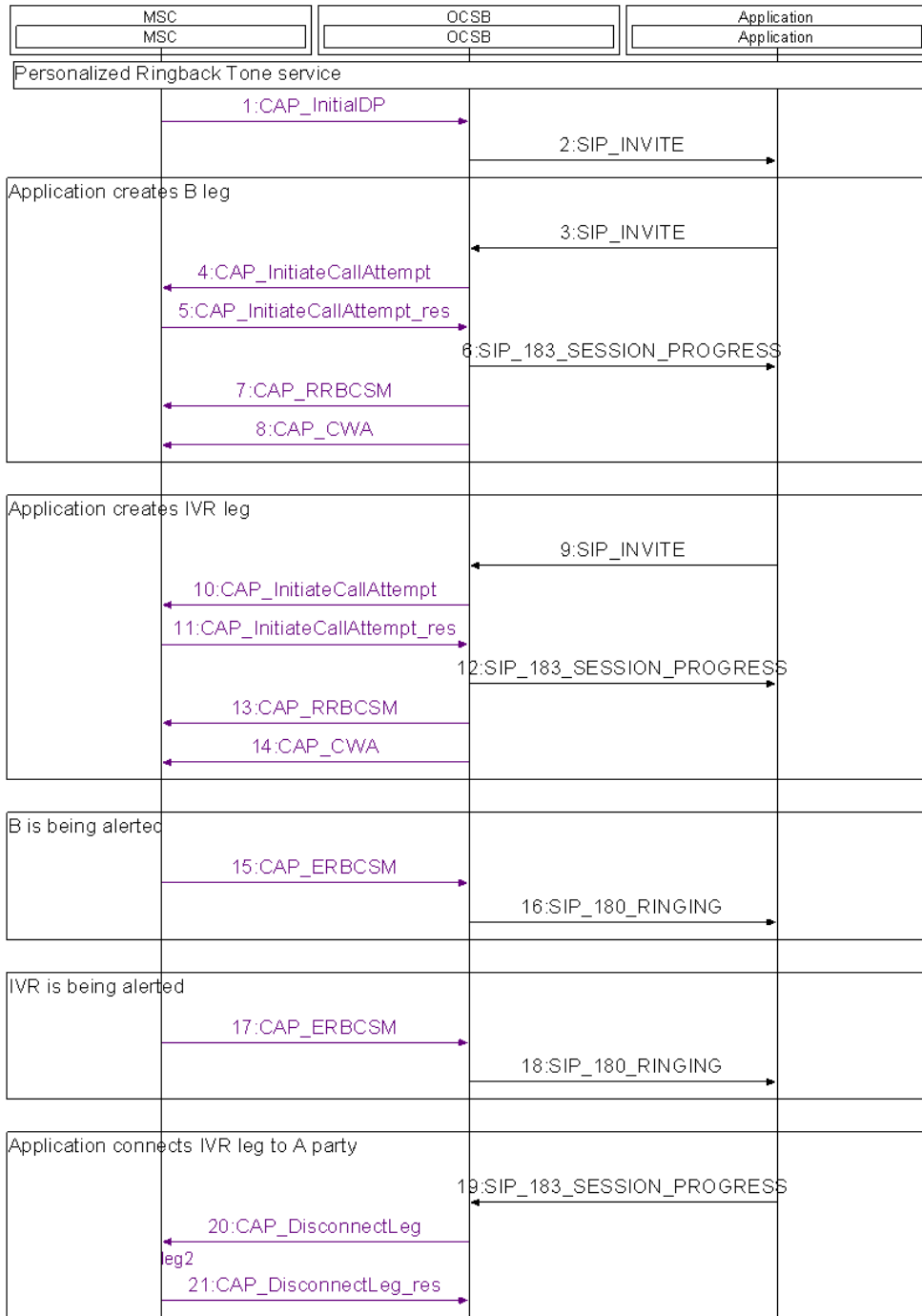


Figure A-2 Personalized Ring Back Tone (cont'd)

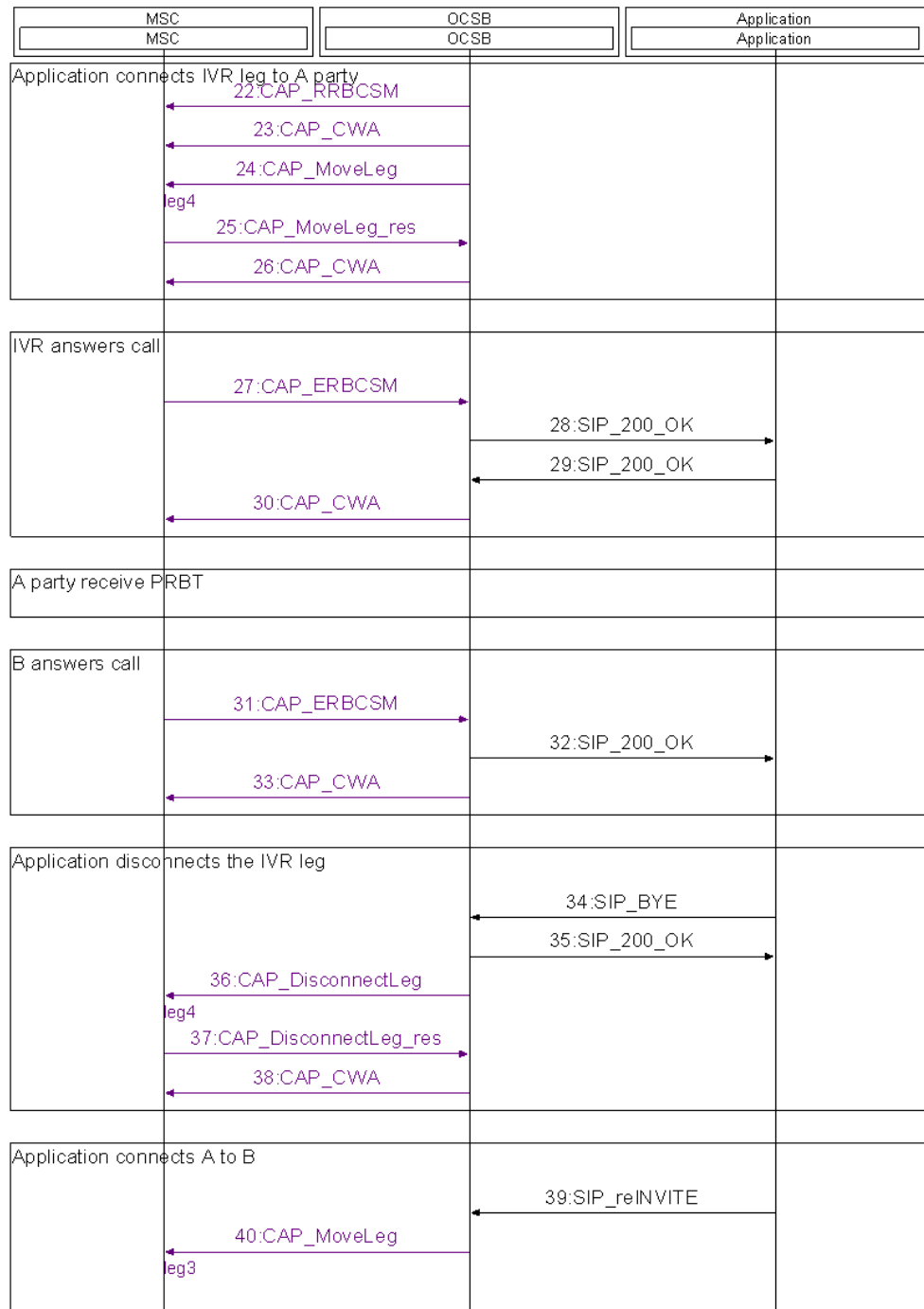
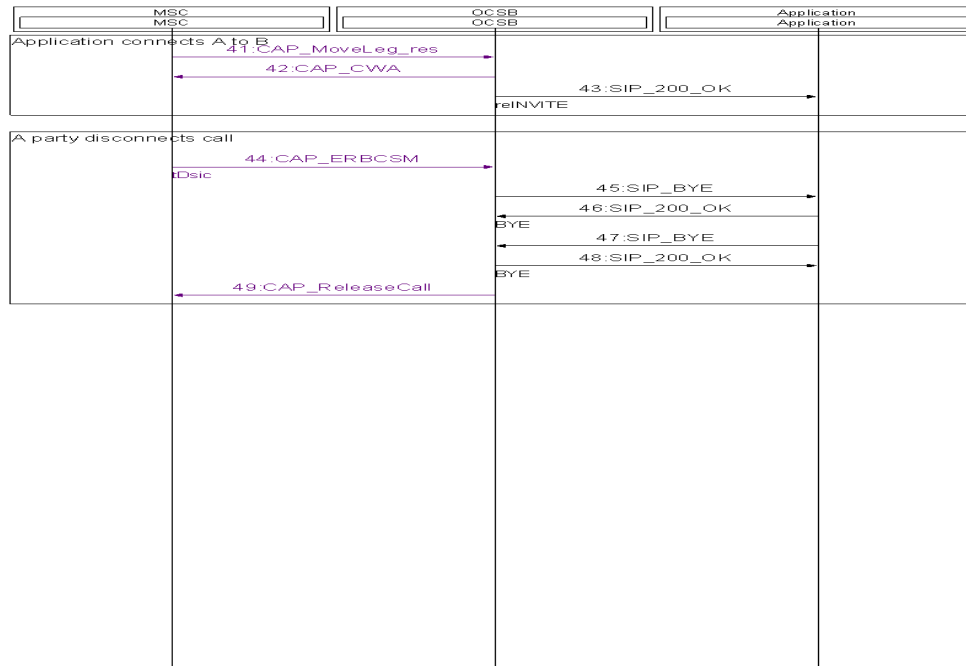


Figure A-3 Personalized Ring Back Tone (cont'd)



Message Dumps

This section provides dumps of messages shown on [Figure A-1](#), [Figure A-2](#), and [Figure A-3](#).

Message #1: CAP InitialDP

Direction: MSC -> Service Broker

Message Content

```
serviceKey: 24
calledPartyNumber: 8490630379003002
  1... .... = Odd/even indicator: odd number of address signals
  .000 0100 = Nature of address indicator: international number (4)
  1... .... = INN indicator: routing to internal network number not allowed
  .001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
Called Party Number: 36309700032
callingPartyNumber: 83130343430508
  1... .... = Odd/even indicator: odd number of address signals
  .000 0011 = Nature of address indicator: national (significant) number (3)
  0... .... = NI indicator: complete
  .001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
  .... 00.. = Address presentation restricted indicator: presentation allowed (0)
  .... ..11 = Screening indicator: network provided (3)
Calling Party Number: 303434508
callingPartysCategory: ordinary calling subscriber (10)
locationNumber: 84936303000000009000
bearerCapability: bearerCap (0)
eventTypeBCSM: termAttemptAuthorized (12)
imsi: 12364010539261F7
TBCD digits: 216304013529167
ext-basicServiceCode: ext-Teleservice (3)
callReferenceNumber: 0000210078
mscAddress: 916303499889F0
timeAndTimezone: 0201306181637200
initialDPArgExtension
```

Message #2: SIP INVITE

Direction: Service Broker -> Application

Message Headers

```
Request-Line: INVITE sip:+36309700032@wcs.convergin.com:5085 SIP/2.0
CSeq: 1 INVITE
Supported: 100rel
x-wcs-msc-address: 916303499889F0
Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Route: <sip:as@192.168.1.112:5060;term;lr>
Route:
<sip:scim@192.168.1.11:5085;lr;appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;xwcs-iteration=2;term>
Route:
<sip:imscf@wcs.convergin.com:5085;appsessionid=app-16nopifvb95i:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;term;lr>
x-wcs-network-name: imscf
x-wcs-service-key: 24
Via: SIP/2.0/TCP 192.168.1.11:5085;wlsscid=17c775b61;maddr=192.168.1.11;branch=z9hG4bKad66972d50042edfed7142eeclaaa2ef
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
Content-Type: multipart/mixed;boundary="frontier"
To: sip:+36309700032@wcs.convergin.com:5085
CPC: ordinary
Contact: <sip:192.168.1.11:5085;transport=udp;wlsscid=17c775b61;appsessionid=app-kqfvpbh7lnp4>
Content-Length: 1108
x-wcs-session-case: term
Subject: call control
x-wcs-mobile-number: 216304013529167
Max-Forwards: 70
```

Message Body

```
--frontier
Content-Type: application/cap-phase4+xml

<?xml version="1.0" encoding="UTF-8"?>
<Cap4>
  <initialDP>
    <serviceKey>24</serviceKey>
    <calledPartyNumber>84906303 79003002</calledPartyNumber>
    <callingPartyNumber>83130343 430508</callingPartyNumber>
    <callingPartysCategory>0A</callingPartysCategory>
    <locationNumber>84936303 00000000 9000</locationNumber>
    <bearerCapability>
      <bearerCap>8090A3</bearerCap>
    </bearerCapability>
    <eventTypeBCSM><termAttemptAuthorized/></eventTypeBCSM>
    <imsi>12364010 539261F7</imsi>
    <ext-basicServiceCode>
      <ext-Teleservice>11</ext-Teleservice>
    </ext-basicServiceCode>
    <callReferenceNumber>00002100 78</callReferenceNumber>
    <mscAddress>91630349 9889F0</mscAddress>
    <timeAndTimezone>02013061 81637200</timeAndTimezone>
```

```
<initialDPArgExtension>  
  <supportedCamelPhases>1111</supportedCamelPhases>  
  <offeredCamel4Functionalities>11111101 11110011  
1</offeredCamel4Functionalities>  
  </initialDPArgExtension>  
</initialDP>  
</Cap4>
```

```
--frontier  
Content-Type: application/sdp  
i=L1
```

```
--frontier--
```

Message #3: SIP_INVITE

Direction: Application -> Service Broker

Message Headers

```
Request-Line: INVITE sip:+36305262266@wcs.convergin.com SIP/2.0
CPC: ordinary
To: sip:+36305262266@wcs.convergin.com
Content-Length: 21
Contact:
<sip:192.168.1.112:5080;transport=udp;wlsscid=-20f515011da29e9a;appsessionid=
app-r7ylctikob20>
CSeq: 1 INVITE
Supported: 100rel
Route:
<sip:scim@192.168.1.11:5085;lr;appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77
c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;xwcs-iteration=2;term>
Route:
<sip:imscf@wcs.convergin.com:5085;appsessionid=app-16nopifvb95i:b4c492e7845110ab25
c77c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;term;lr>
Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK00feaa55fc949977402036ba6738f261;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e
Content-Type: application/sdp
Max-Forwards: 70
```

Message Body

```
Session Description Protocol
Session Information (i): 0
Connection Information (c): IN IP4 0.0.0.0
```

Message #4: CAP InitiateCallAttempt

Direction: Service Broker -> MSC

Message Content

```
destinationRoutingAddress: 1 item
  Item: 8490630325266206
    1... .... = Odd/even indicator: odd number of address signals
    .000 0100 = Nature of address indicator: international number (4)
    1... .... = INN indicator: routing to internal network number not allowed
    .001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
    Called Party Number: 36305262266
legToBeCreated: sendingSideID (0)
  sendingSideID: 03
newCallSegment: 2
callingPartyNumber: 83130343430508
1... .... = Odd/even indicator: odd number of address signals
.000 0011 = Nature of address indicator: national (significant) number (3)
0... .... = NI indicator: complete
.001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
.... 00.. = Address presentation restricted indicator: presentation allowed (0)
.... ..11 = Screening indicator: network provided (3)
Calling Party Number: 303434508
gsmSCFAddress: 916303899740F9
```

Message #5: CAP InitiateCallAttempt Result

Direction: MSC -> Service Broker

Message Content

```
Padding: 4
supportedCamelPhases: F0 (phase1, phase2, phase3, phase4)
1... .... = phase1: True
.1.. .... = phase2: True
..1. .... = phase3: True
...1 .... = phase4: True
Padding: 4
offeredCamel4Functionalities: FDF380 (initiateCallAttempt, splitLeg, moveLeg,
disconnectLeg, entityReleased, dfc-WithArgument, dtmf-MidCall, chargingIndicator,
alertingDP, locationAtAlerting, changeOfPositionDP, cf-Enhancements, subscribed)
1... .... = initiateCallAttempt: True
.1.. .... = splitLeg: True
..1. .... = moveLeg: True
...1 .... = disconnectLeg: True
.... 1... = entityReleased: True
.... .1.. = dfc-WithArgument: True
.... ..0. = playTone: False
.... ...1 = dtmf-MidCall: True
1... .... = chargingIndicator: True
.1.. .... = alertingDP: True
..1. .... = locationAtAlerting: True
...1 .... = changeOfPositionDP: True
.... 0... = or-Interactions: False
.... .0.. = warningToneEnhancements: False
.... ..1. = cf-Enhancements: True
.... ...1 = subscribedEnhancedDialledServices: True
1... .... = servingNetworkEnhancedDialledServices: True
.0.. .... = criteriaForChangeOfPositionDP: False
..0. .... = serviceChangeDP: False
...0 .... = collectInformation: False
```

Message #6: SIP 183 SESSION PROGRESS

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 183 Session Progress
To: <sip:+36305262266@wcs.convergin.com>;tag=309f5d2b
Content-Length: 204
CSeq: 1 INVITE
Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK00feaa55fc949977402036ba6738f261;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e
Content-Type: multipart/mixed;boundary="frontier"
x-wcs-encode-uri:
<sip:imscf@wcs.convergin.com:5085;appsessionid=app-16nopifvb95i:b4c492e7845110ab25
c77c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;lr>
x-wcs-encode-uri:
<sip:scim@192.168.1.11:5085;lr;appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77
c25ea47da4e%4010.107.14.31;wlsscid=17c775b61>
```

Message Body

```
--frontier
Content-Type: application/cap-phase4+xml
<?xml version="1.0" encoding="UTF-8"?>
<Cap4>
  <initiateCallAttemptResult/>
</Cap4>

--frontier
Content-Type: application/sdp
i=L3

--frontier-
```

Message #7: CAP RequestReportBCSMEvent

Direction: Service Broker -> MSC

Message Content

```
bcsmEvents: 6 items
  Item
    eventTypeBCSM: routeSelectFailure (4)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 03
  Item
    eventTypeBCSM: oDisconnect (9)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 03
  Item
    eventTypeBCSM: oAnswer (7)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 03
  Item
    eventTypeBCSM: oTermSeized (19)
    monitorMode: notifyAndContinue (1)
    legID: sendingSideID (0)
    sendingSideID: 03
  Item
    eventTypeBCSM: oNoAnswer (6)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 03
  Item
    eventTypeBCSM: oCalledPartyBusy (5)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 03
```


Message #8: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: legID (1)  
    legID: sendingSideID (0)  
      sendingSideID: 03
```

Message #9: SIP INVITE

Direction: Application -> Service Broker

Message Headers

```
Request-Line: INVITE sip:+3612911104@wcs.convergin.com SIP/2.0
CPC: ordinary
To: sip:+3612911104@wcs.convergin.com
Content-Length: 21
Contact:
<sip:192.168.1.112:5080;transport=udp;wlsscid=-20f515011da29e9a;appsessionid=
app-r7ylctikob20>
CSeq: 1 INVITE
Supported: 100rel
Route:
<sip:scim@192.168.1.11:5085;lr;appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77
c25ea47da4e%
4010.107.14.31;wlsscid=17c775b61;xwcs-iteration=2;term>
Route:
<sip:imscf@wcs.convergin.com:5085;appsessionid=app-16nopifvb95i:b4c492e7845110ab25
c77c25ea47da4e%
4010.107.14.31;wlsscid=17c775b61;term;lr>
Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK6f7027cb5e1149ebdad1c50d040af180;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
Content-Type: application/sdp
Max-Forwards: 70
```

Message Body

```
Session Description Protocol
Session Information (i): 0
Connection Information (c): IN IP4 0.0.0.0
```

Message #10: CAP InitiateCallAttempt

Direction: Service Broker -> MSC

Message Content

```
destinationRoutingAddress: 1 item
  Item: 04906321191140
    0... .... = Odd/even indicator: even number of address signals
    .000 0100 = Nature of address indicator: international number (4)
    1... .... = INN indicator: routing to internal network number not allowed
    .001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
Called Party Number: 3612911104
legToBeCreated: sendingSideID (0)
  sendingSideID: 04
newCallSegment: 3
callingPartyNumber: 83130343430508
1... .... = Odd/even indicator: odd number of address signals
.000 0011 = Nature of address indicator: national (significant) number (3)
0... .... = NI indicator: complete
.001 .... = Numbering plan indicator: ISDN (Telephony) numbering plan (1)
.... 00.. = Address presentation restricted indicator: presentation allowed (0)
.... ..11 = Screening indicator: network provided (3)
Calling Party Number: 303434508
gsmSCFAddress: 916303899740F9
```

Message #11: CAP InitiateCallAttempt Result

Direction: MSC -> Service Broker

Message Content

```
supportedCamelPhases: F0 (phase1, phase2, phase3, phase4)
  1... .... = phase1: True
  .1.. .... = phase2: True
  ..1. .... = phase3: True
  ...1 .... = phase4: True
Padding: 4
offeredCamel4Functionalities: FDF380 (initiateCallAttempt, splitLeg, moveLeg,
disconnectLeg, entityReleased, dfc-WithArgument, dtmf-MidCall, chargingIndicator,
alertingDP, locationAtAlerting, changeOfPositionDP, cf-Enhancements, subscribed
  1... .... = initiateCallAttempt: True
  .1.. .... = splitLeg: True
  ..1. .... = moveLeg: True
  ...1 .... = disconnectLeg: True
  .... 1... = entityReleased: True
  .... .1.. = dfc-WithArgument: True
  .... ..0. = playTone: False
  .... ...1 = dtmf-MidCall: True
  1... .... = chargingIndicator: True
  .1.. .... = alertingDP: True
  ..1. .... = locationAtAlerting: True
  ...1 .... = changeOfPositionDP: True
  .... 0... = or-Interactions: False
  .... .0.. = warningToneEnhancements: False
  .... ..1. = cf-Enhancements: True
  .... ...1 = subscribedEnhancedDialledServices: True
  1... .... = servingNetworkEnhancedDialledServices: True
  .0.. .... = criteriaForChangeOfPositionDP: False
  ..0. .... = serviceChangeDP: False
  ...0 .... = collectInformation: False
```

Message #12: SIP 183 SESSION PROGRESS

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 183 Session Progress
  To: <sip:+3612911104@wcs.convergin.com>;tag=b005d41d
  Content-Length: 204
  CSeq: 1 INVITE
  Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK6f7027cb5e1149ebdad1c50d040af180;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
  From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
  Content-Type: multipart/mixed;boundary="frontier"
  x-wcs-encode-uri:
<sip:imscf@wcs.convergin.com:5085;appsessionid=app-16nopifvb95i:b4c492e7845110ab25
c77c25ea47da4e%4010.107.14.31;wlsscid=17c775b61;lr>
  x-wcs-encode-uri:
<sip:scim@192.168.1.11:5085;lr;appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77
c25ea47da4e%4010.107.14.31;wlsscid=17c775b61>
```

Message Body

```
--frontier
Content-Type: application/cap-phase4+xml
<?xml version="1.0" encoding="UTF-8"?>
<Cap4>
  <initiateCallAttemptResult/>
</Cap4>

--frontier
Content-Type: application/sdp
i=L4

--frontier-
```

Message #13: CAP RequestReportBCSMEvent

Direction: Service Broker -> MSC

Message Content

```
bcsmEvents: 6 items
  Item
    eventTypeBCSM: routeSelectFailure (4)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 04
  Item
    eventTypeBCSM: oDisconnect (9)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 04
  Item
    eventTypeBCSM: oAnswer (7)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 04
  Item
    eventTypeBCSM: oTermSeized (19)
    monitorMode: notifyAndContinue (1)
    legID: sendingSideID (0)
    sendingSideID: 04
  Item
    eventTypeBCSM: oNoAnswer (6)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 04
  Item
    eventTypeBCSM: oCalledPartyBusy (5)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 04
```

Message #14: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: legID (1)  
    legID: sendingSideID (0)  
      sendingSideID: 04
```

Message #15: CAP EventReportBCSM

Direction: MSC -> Service Broker

Message Content

```
eventTypeBCSM: oTermSeized (19)
legID: receivingSideID (1)
  receivingSideID: 03
miscCallInfo
  messageType: notification (1)
```


Message #16: SIP 180 RINGING

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 180 Ringing
To: <sip:+36305262266@wcs.convergin.com>;tag=309f5d2b
Content-Length: 0
CSeq: 1 INVITE
Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK00feaa55fc949977402036ba6738f261;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e
Content-Type: multipart/mixed;boundary="frontier"
```

Message Body

n/a

Message #17: CAP EventReportBCSM

Direction: MSC -> Service Broker

Message Content

```
eventTypeBCSM: oTermSeized (19)
legID: receivingSideID (1)
    receivingSideID: 04
miscCallInfo
    messageType: notification (1)
```

Message #18: SIP 180 RINGING

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 180 Ringing
To: <sip:+3612911104@wcs.convergin.com>;tag=b005d41d
Content-Length: 0
CSeq: 1 INVITE
Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK6f7027cb5e1149ebdad1c50d040af180;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
```

Message Body

n/a

Message #19: SIP 183 SESSION PROGRESS

Direction: Application -> Service Broker

Message Headers

```
Status-Line: SIP/2.0 183 Session Progress
To: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
Content-Length: 4
CSeq: 1 INVITE
Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/TCP 192.168.1.11:5085;wlsscid=17c775b61;maddr=192.168.1.11;branch=
z9hG4bKad66972d50042edfed7142eec1aaa2ef;received=10.107.14.11
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
Content-Type: application/sdp
```

Message Body

```
Session Description Protocol
  Session Information (i): L4
```

Message #20: CAP DisconnectLeg

Direction: Service Broker -> MSC

Message Content

```
legToBeReleased: sendingSideID (0)
  sendingSideID: 02
releaseCause: 809F
  .... 0000 = Cause location: User (U) (0)
  .00. .... = Coding standard: ITU-T standardized coding (0x00)
  1... .... = Extension indicator: last octet
  .001 1111 = Cause indicator: Normal unspecified (31)
  1... .... = Extension indicator: last octet
```

Message #21: CAP DisconnectLeg Result

Direction: MSC -> Service Broker

Message Content

```
returnResult
  invokeId: present (0)
  present: 6
```

Message #22: CAP RequestReportBCSM

Direction: Service Broker -> MSC

Message Content

```
bcsmEvents: 2 items
  Item
    eventTypeBCSM: tAbandon (18)
    monitorMode: interrupted (0)
  Item
    eventTypeBCSM: tDisconnect (17)
    monitorMode: interrupted (0)
    legID: sendingSideID (0)
    sendingSideID: 01
```

Message #23: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: callSegmentID (0)  
  callSegmentID: 1
```


Message #24: CAP MoveLeg

Direction: Service Broker -> MSC

Message Content

```
legIDToMove: sendingSideID (0)
  sendingSideID: 04
```

Message #25: CAP MoveLeg Result

Direction: MSC -> Service Broker

Message Content

```
returnResult
  invokeId: present (0)
  present: 9
```

Message #26: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
legOrCallSegment: legID (1)
  legID: sendingSideID (0)
    sendingSideID: 04
```

Message #27: CAP EventReportBCSM

Direction: MSC -> Service Broker

Message Content

```
eventTypeBCSM: oAnswer (7)
eventSpecificInformationBCSM: oAnswerSpecificInfo (5)
legID: receivingSideID (1)
    receivingSideID: 04
miscCallInfo
    messageType: request (0)
```

Message #28: SIP 200 OK**Direction:** Service Broker -> Application**Message Headers**

```
Status-Line: SIP/2.0 200 OK
  To: <sip:+3612911104@wcs.convergin.com>;tag=b005d41d
  Contact:
<sip:app-kqfvpbh7lnp4@192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
  appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31>
  Content-Length: 649
  CSeq: 1 INVITE
  Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
  branch=z9hG4bK6f7027cb5e1149ebdad1c50d040af180;appsessionid="app-kqfvpbh7lnp4:b4c4
  92e7845110ab25c77c25ea47da4e@10.107.14.31"
  From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
  Content-Type: multipart/mixed;boundary="frontier"
```

Message Body

```
--frontier
Content-Type: application/cap-phase4+xml
<?xml version="1.0" encoding="UTF-8"?>
<Cap4>
  <eventReportBCSM>
    <eventTypeBCSM><oAnswer/></eventTypeBCSM>
    <eventSpecificInformationBCSM>
      <oAnswerSpecificInfo>
        <destinationAddress>03902119 1140</destinationAddress>
        <chargeIndicator>02</chargeIndicator>
      </oAnswerSpecificInfo>
    </eventSpecificInformationBCSM>
    <legID>
      <receivingSideID>04</receivingSideID>
    </legID>
    <miscCallInfo>
      <messageType><request/></messageType>
    </miscCallInfo>
  </eventReportBCSM>
</Cap4>

--frontier
Content-Type: application/sdp
i=L4

--frontier--
```

Message #29: SIP 200 OK

Direction: Application -> Service Broker

Message Headers

```
Status-Line: SIP/2.0 200 OK
To: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
Contact: <sip:app-r7ylctikob20@192.168.1.112:5060;transport=tcp;wlsscid=-20f515011da29e9a>
Content-Length: 4
CSeq: 1 INVITE
Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/TCP 192.168.1.11:5085;wlsscid=17c775b61;maddr=192.168.1.11;branch=z9hG4bKad66972d50042edfed7142eec1aaa2ef;received=10.107.14.11
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
Content-Type: application/sdp
```

Message Body

```
Session Description Protocol
Session Information (i): L4
```

Message #30: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: legID (1)  
    legID: sendingSideID (0)  
      sendingSideID: 04
```

Message #31: CAP EventReportBCSM

Direction: MSC -> Service Broker

Message Content

```
eventTypeBCSM: oAnswer (7)
eventSpecificInformationBCSM: oAnswerSpecificInfo (5)
legID: receivingSideID (1)
    receivingSideID: 03
miscCallInfo
    messageType: request (0)
```

Message #32: SIP 200 OK

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 200 OK
To: <sip:+36305262266@wcs.convergin.com>;tag=309f5d2b
Contact:
<sip:app-kqfvpbh7lnp4@192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
  appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31>
Content-Length: 653
CSeq: 1 INVITE
Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK00feaa55fc949977402036ba6738f261;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e
Content-Type: multipart/mixed;boundary="frontier"
```

Message Body

```
MIME Multipart Media Encapsulation, Type: multipart/mixed, Boundary: "frontier"
[Type: multipart/mixed]
Preamble
First boundary: --frontier\n
Encapsulated multipart part:
[Malformed Packet: MIME multipart]
```

Message #33: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: callSegmentID (0)  
  callSegmentID: 1
```

Message #34: SIP BYE

Direction: Application -> Service Broker

Message Headers

```
Request-Line: BYE
sip:app-kqfvpbh7lnp4@192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31
SIP/2.0
  To: <sip:+3612911104@wcs.convergin.com>;tag=b005d41d
  Content-Length: 0
  CSeq: 2 BYE
  Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK492e7fb11d5aa895cf427d70f70fe55d;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
  From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
  Max-Forwards: 70
```

Message Body

n/a

Message #35: SIP 200 OK

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 200 OK
To: <sip:+3612911104@wcs.convergin.com>;tag=b005d41d
Content-Length: 0
CSeq: 2 BYE
Call-ID: wlss-1657829b-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bK492e7fb11d5aa895cf427d70f70fe55d;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=c706ccc6
```

Message Body

n/a

Message #36: CAP DisconnectLeg

Direction: Service Broker -> MSC

Message Content

```
legToBeReleased: sendingSideID (0)
  sendingSideID: 04
releaseCause: 809F
  .... 0000 = Cause location: User (U) (0)
  .00. .... = Coding standard: ITU-T standardized coding (0x00)
  1... .... = Extension indicator: last octet
  .001 1111 = Cause indicator: Normal unspecified (31)
  1... .... = Extension indicator: last octet
```

Message #37: CAP DisconnectLeg Result

Direction: MSC -> Service Broker

Message Content

```
returnResult
  invokeId: present (0)
  present: 13
```

Message #38: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: callSegmentID (0)  
  callSegmentID: 1
```

Message #39: SIP reINVITE

Direction: Application -> Service Broker

Message Headers

```
Request-Line: INVITE sip:192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
appsessionid=app-kqfvpbh7lnp4 SIP/2.0
To: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
Content-Length: 4
Contact: <sip:192.168.1.112:5060;transport=udp;wlsscid=-20f515011da29e9a;
appsessionid=app-r7ylctikob20>
CSeq: 1 INVITE
Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bKcdf7560c104fa752004bebca0bb7f19f
From: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
Content-Type: application/sdp
Max-Forwards: 70
```

Message Body

```
Session Description Protocol
Session Information (i): L3
```


Message #40: CAP MoveLeg

Direction: Service Broker -> MSC

Message Content

```
legIDToMove: sendingSideID (0)  
sendingSideID: 03
```

Message #41: CAP MoveLeg Result

Direction: MSC -> Service Broker

Message Content

```
returnResult
  invokeId: present (0)
  present: 15
```

Message #42: CAP ContinueWithArgument

Direction: Service Broker -> MSC

Message Content

```
continueWithArgumentArgExtension  
  legOrCallSegment: callSegmentID (0)  
  callSegmentID: 1
```

Message #43: SIP 200 OK

Direction: Service Broker -> Application

Message Headers

```
Status-Line: SIP/2.0 200 OK
  To: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
  Content-Length: 0
  Contact:
<sip:app-kqfvpbh7lnp4@192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
  appsessionid=app-kqfvpbh7lnp4>
  CSeq: 1 INVITE
  Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
  branch=z9hg4bKcdf7560c104fa752004bebca0bb7f19f
  From: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
```

Message Body

n/a

Message #44: CAP EventReportBCSM

Direction: MSC -> Service Broker

Message Content

```
eventTypeBCSM: tDisconnect (17)
eventSpecificInformationBCSM: tDisconnectSpecificInfo (12)
legID: receivingSideID (1)
    receivingSideID: 01
miscCallInfo
    messageType: request (0)
```

Message #45: SIP BYE

Direction: Service Broker -> Application

Message Headers

```
Request-Line: BYE
sip:192.168.1.112:5060;transport=udp;wlsscid=-20f515011da29e9a;appsessionid=app-r7
ylctikob20 SIP/2.0
  To: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
  Reason: Q.850;cause=16;text=NORMAL_CALL_CLEARING
  Content-Length: 594
  CSeq: 2 BYE
  Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.11:5085;wlsscid=17c775b61;maddr=192.168.1.11;
  branch=z9hg4bK14e1c74fb81777ae220871fbc535f3d3
  From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
  Content-Type: multipart/mixed;boundary="frontier"
  Max-Forwards: 70
```

Message Body

n/a

Message #46: SIP 200 OK

Direction: Application -> Service Broker

Message Headers

```
Status-Line: SIP/2.0 200 OK
To: <sip:+36309700032@wcs.convergin.com:5085>;tag=4479eccf
Content-Length: 0
CSeq: 2 BYE
Call-ID: wlss-95022050-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.11:5085;wlsscid=17c775b61;maddr=192.168.1.11;
branch=z9hG4bK14e1c74fb81777ae220871fbc535f3d3
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=70c65f8a
```

Message Body

n/a

Message #47: SIP BYE

Direction: Application -> Service Broker

Message Headers

```
Request-Line: BYE
sip:app-kqfvpbh7lnp4@192.168.1.11:5085;transport=udp;wlsscid=17c775b61;
appsessionid=app-kqfvpbh7lnp4:b4c492e7845110ab25c77c25ea47da4e%4010.107.14.31
SIP/2.0
  To: <sip:+36305262266@wcs.convergin.com>;tag=309f5d2b
  Content-Length: 594
  CSeq: 2 BYE
  Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
  Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bKf68564a3a09538c236130273b0f7e125;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
  From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e
  Content-Type: multipart/mixed;boundary="frontier"
  Max-Forwards: 70
```

Message Body

```
MIME Multipart Media Encapsulation, Type: multipart/mixed, Boundary: "frontier"
[Type: multipart/mixed]
Preamble
First boundary: --frontier\n
Encapsulated multipart part:
[Malformed Packet: MIME multipart]
```


Message #48: SIP 200 OK

Direction: Service Broker -> Application

Message Headers

Status-Line: SIP/2.0 200 OK
To: <sip:+36305262266@wcs.convergin.com>;tag=309f5d2b
Content-Length: 0
CSeq: 2 BYE
Call-ID: wlss-3c6798d2-b4c492e7845110ab25c77c25ea47da4e@10.107.14.31
Via: SIP/2.0/UDP 192.168.1.112:5060;wlsscid=-20f515011da29e9a;
branch=z9hG4bKf68564a3a09538c236130273b0f7e125;appsessionid="app-kqfvpbh7lnp4:b4c4
92e7845110ab25c77c25ea47da4e@10.107.14.31"
From: <sip:303434508@wcs.convergin.com:5085;noa=national>;tag=9b45dc4e

Message Body

n/a

Message #49: CAP ReleaseCall

Direction: Service Broker -> MSC

Message Content

```
.... 0000 = Cause location: User (U) (0)
.00. .... = Coding standard: ITU-T standardized coding (0x00)
1... .... = Extension indicator: last octet
.001 1111 = Cause indicator: Normal unspecified (31)
1... .... = Extension indicator: last octet
```