ETSI TR 129 998-4-4 V7.0.0 (2007-03)

Technical Report

Universal Mobile Telecommunications System (UMTS); Open Service Access (OSA) Application Programming Interface (API) Mapping for Open Service Access; Part 4: Call Control Service Mapping; Subpart 4: Multiparty Call Control ISC (3GPP TR 29.998-04-4 version 7.0.0 Release 7)

GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS

Reference
RTS/TSGC-0529998-04-4v700

Keywords
GSM, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

Structure of the OSA API Mapping (3GPP TR 29.998)

The present document is part 4, subpart 4, of a multi-part deliverable covering the Open Service Access (OSA); Application Programming Interface (API) Mapping for OSA.

Table: Overview of the OSA APIs & Protocol Mappings 29.198 & 29.998-family

OSA API specifications 29.198-family		OSA API Mapping - 29.998-family	
29.198-01	Overview	29.998-01	Overview
29.198-02	Common Data Definitions	*29.998-02*	*Not Applicable*
29.198-03	Framework	*29.998-03*	*Not Applicable*
Call Control (CC) SCF	29.198-04-1 Common CC data definitions	29.998-04-1	Generic Call Control – CAP mapping
	29.198-04-2 Generic CC SCF	*29.998-04-2*	*Generic Call Control – INAP mapping*
	29.198-04-3 Multi-Party CC SCF	*29.998-04-3*	*Generic Call Control – Megaco mapping*
	29.198-04-4 Multi-media CC SCF	**29.998-04-4**	**Multiparty Call Control – ISC mapping**
	29.198-04-5 Conf. CC SCF		
29.198-05	User Interaction SCF	29.998-05-1	User Interaction – CAP mapping
		29.998-05-2	*User Interaction – INAP mapping*
		29.998-05-3	*User Interaction – Megaco mapping*
		29.998-05-4	User Interaction – SMS mapping
29.198-06	Mobility SCF	29.998-06-1	User Status and User Location – MAP mapping
		29.998-06-2	User Status and User Location – SIP mapping
29.198-07	Terminal Capabilities SCF	*29.998-07*	*Not Applicable*
29.198-08	Data Session Control SCF	29.998-08	Data Session Control – CAP mapping
29.198-09	*Generic Messaging SCF*	*29.998-09*	*Not Applicable*
29.198-10	*Connectivity Manager SCF*	*29.998-10*	*Not Applicable*
29.198-11	Account Management SCF	*29.998-11*	*Not Applicable*
29.198-12	Charging SCF	*29.998-12*	*Not Applicable*
29.198-13	Policy Management SCF	*29.998-13*	*Not Applicable*
29.198-14	Presence & Availability Management SCF	*29.998-14*	*Not Applicable*
29.198-15	Multi Media Messaging SCF	*29.998-15*	*Not Applicable*
29.198-16	Service Broker SCF	*29.998-16*	*Not Applicable*

1 Scope

The present document investigates how the OSA Call Control Interface Class methods defined in [5] can be mapped onto SIP methods.
The mapping of the OSA API to the SIP is considered informative, and not normative. An overview of the mapping TR is contained in the introduction of the present document as well as in 3GPP TR 29.998-1 [10].

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs. The API specification is contained in the 3GPP TS 29.198 series of specifications. An overview of these is available in the introduction of the present document as well as in 3GPP TS 29.198-1 [1]. The concepts and the functional architecture for the Open Service Access (OSA) are described by 3GPP TS 23.198 [3]. The requirements for OSA are defined in 3GPP TS 22.127 [2].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 29.198-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview".

[2] 3GPP TS 22.127: "Service Requirement for the Open Service Access (OSA); Stage 1".

[3] 3GPP TS 23.198: "Open Service Access (OSA); Stage 2".

[4] 3GPP TR 21.905: "Vocabulary for 3GPP specifications".

[5] 3GPP TS 29.198-4-1/5: "Open Service Access (OSA); Application Programming Interface (API); Part 4: Call control; Sub-part 1: Call Control Common Definitions".
Sub-part 2: Generic Call Control SCF".
Sub-part 3: "Multi-Party Call Control SCF".
Sub-part 4: "Multi-Media Call Control SCF".
Sub-part 5: "Conference call control SCF".

[6] 3GPP TS 23.218: "IP Multimedia (IM) session handling; IP Multimedia (IM) call model; Stage 2".

[7] 3GPP TS 22.101: "Service aspects; Service principles".

[8] 3GPP TS 29.228 " IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents".

[9] 3GPP TR 29.998-1: " Open Service Access (OSA); Application Programming Interface (API) Mapping for Open Service Access; Part 1: General Issues on API Mapping".

[10] IETF RFC 2806: "URLs for Telephone Calls".

[11] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2".

[12] 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on SIP and SDP; Stage 3".

[13] 3GPP TS 24.228: "Signalling flows for the IP multimedia call control based on SIP and SDP; Stage 3".

[14] RFC 3261: "SIP: Session Initiation Protocol".

[15] 3GPP TS 29.328: "IP Multimedia Subsystem (IMS) Sh Interface signalling flows and message contents".

[16] RFC 3725: "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.198-1 [1], TS 23.228 [11] and TS 24.228 [13] and the following apply:

Back-To-Back User Agent (B2BUA): logical entity that receives a request, and processes it as a UAS
In order to determine how the request should be answered, it acts as a UAC and generates requests. Unlike a proxy server, it maintains dialog state, and must participate in all requests sent on the dialogs it has established. Since it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behaviour.

call: informal term that refers to a dialog between peers, generally set up for the purposes of a multimedia conversation

call leg: another name for a dialogue in a SIP context
In an OSA context the communication path as seen from an application to an addressable entity/call party in the network.

call stateful: proxy which retains state for a dialog from the initiating INVITE to the terminating BYE request

client: any network element that sends SIP requests, and receives SIP responses
Clients may or may not interact directly with a human user. User agent clients and proxies are clients.

dialog: peer-to-peer SIP relationship between a UAC and UAS that persists for some time
A dialog is established by SIP messages, such as a 2xx response to an INVITE request. A dialog is identified by a call identifier, local address, and remote address.

downstream: direction of message forwarding within a transaction which refers to the direction that requests flow from the user agent client to user agent server

final response: response that terminates a SIP transaction, as opposed to a provisional response that does not
All 2xx, 3xx, 4xx, 5xx and 6xx responses are final.

informational response: provisional response

initiator, calling party, caller: The party initiating a session with an INVITE request. A caller retains this role from the time it sends the INVITE until the termination of any dialogs established by the INVITE.

invitation: INVITE request.

invitee, invited user, called party, callee: party that receives an INVITE request for the purposes of establishing a new session. A callee retains this role from the time it receives the INVITE until the termination of the dialog established by that INVITE.

location server: See location service.

location service: service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s)
It is an abstract database, sometimes referred to as a location server. The contents of the database can be populated in many ways, including being written by registrars.

method: primary function that a request is meant to invoke on a server
The method is carried in the request message itself. Example methods are INVITE and BYE.

outbound proxy: proxy that receives all requests from a client, even though it is not the server resolved by the Request-URI
The outbound proxy sends these requests, after any local processing, to the address indicated in the Request-URI, or to another outbound proxy.

parallel search: In a parallel search, a proxy issues several requests to possible user locations upon receiving an incoming request. Rather than issuing one request and then waiting for the final response before issuing the next request as in a sequential search , a parallel search issues requests without waiting for the result of previous requests.

provisional response: response used by the server to indicate progress, but that does not terminate a SIP transaction
1xx responses are provisional, other responses are considered final.

proxy, proxy server: intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients
A proxy server primarily plays to role of routing, which means its job is to ensure that a request is passed on to another entity that can further process the request. Proxies are also useful for enforcing policy and for firewall traversal. A proxy interprets, and, if necessary, rewrites parts of a request message before forwarding it.

redirect server: server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client
Unlike a proxy server, it does not initiate its own SIP request. Unlike a user agent server, it does not accept calls.

registrar: server that accepts REGISTER requests, and places the information it receives in those requests into the location service for the domain it handles

sequential search: in a sequential search, a proxy server attempts each contact address in sequence, proceeding to the next one only after the previous has generated a non-2xx final response

server: network element that receives requests in order to service them, and sends back responses to those requests
Examples of servers are proxies, user agent servers, redirect servers, and registrars.

session: From the SDP specification: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." (see RFC 2327 [6]) (A session as defined for SDP can comprise one or more RTP sessions.) As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the user name , session id , network type , address type and address elements in the origin field.

(SIP) transaction: transaction which occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response sent from the server to the client, and the ACK for the response in the case the response was a 2xx
The ACK for a 2xx response is a separate transaction.

spiral: SIP request which is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but this time, differs in a way which will result in a different processing decision than the original request
Typically, this means that it has a Request-URI that differs from the previous arrival. A spiral is not an error condition, unlike a loop.

stateless proxy: logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests
A stateless proxy forwards every request it receives downstream and every response it receives upstream.

stateful proxy: logical entity that maintains the client and server transaction state machines defined by this specification during the processing of a request
Also known as a transaction stateful proxy.. A stateful proxy is not the same as a call stateful proxy.

upstream: direction of message forwarding within a transaction which refers to the direction that responses flow from the user agent server to user agent client

User Agent Client (UAC): A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later on, it takes on the role of a User Agent Server for the processing of that transaction.

User Agent Server (UAS): logical entity that generates a response to a SIP request
The response accepts, rejects or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later on, it takes on the role of a User agent client for the processing of that transaction.

User Agent (UA): logical entity which can act as both a user agent client and user agent server for the duration of a dialog

user: logical, identifiable entity which uses services
In a SIP context it encompasses a User Agent (UA).

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.198-1 [1] apply.

4 Mapping OSA Call and Call Leg to SIP

4.1 Introduction

In the MPCCS the CallSessionID designates the call as seen from the application, i.e. the ID used to identify a call session. The MPCC API uses this callSessionID to identify a call session.

In SIP, a SIP dialogue (or call) is identified at each UA with a dialog ID, which consists of a Call-ID value, a local tag and a remote tag. by a globally unique call-id. The call-id is created when a user agent sends an INVITE request tries to initiate a dialog. For a UAC, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the To field of the message, and the local tag is set to the tag in the From field of the message (these rules apply to both requests and responses). For a UAS, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the From field of the message, and the local tag is set to the tag in the To field of the message. This INVITE request may generate multiple acceptances, each of which are part of the same call.

However, the semantics of SIP Call-ID is somewhat different from traditional telephony. It identifies an invitation of a particular client. This means that a conference in SIP may raise several calls with different Call-IDs. In traditional telephony and in MPCCS this would always be the same call.

In MPCCS a call leg designates the association between a call and an address as seen from the application and is identified by a callLegSessionID, i.e. the ID used to identify a call leg session. It represents an addressable user in the call. The MPCC API uses this callLegSessionID to identify a call leg session.

In SIP, a dialogue is defined as the pair wise signalling relationship between two SIP user agents (see [13]). It is identified by the **Call_ID, the tags in theTo and From** header Fields. The Call-ID identifies the call in the network. It is a global unique identifier. The To header field contains the information regarding the endpoint who will receive the SIP request, e.g. INVITE or BYE message. The From header field represents the originator of the SIP request.

4.2 SIP Call-id &dialog vs. OSA Call & Call Leg Session ID

There is a correspondence between the concepts Call and Call Leg in OSA and call-ID and dialog in SIP. The correlation applicable depends on the mode (e.g. Proxy, B2BUA, UA) in which the controller (e.g. OSA SCS) operates. When the controller operates in UA mode there can be a simple 1:1 correlation between OSA callLeg and SIP call-ID, in other cases (e.g. when operating in Proxy mode) a somewhat more complex correlation applies that demands supplementary information such as TO and From header fields in SIP to be correlated with the OSA leg identifiers ("callLeg sessionID).

The Call-ID, the From and To header fields define an association between the call (Call-ID) and the address (To, From). Thus we can map the call and call leg concepts in OSA to SIP. However, there is no easy mapping between SIP and OSA MPCCS call and call leg concepts because of the definition of a SIP dialog always include TWO user agents (UAs). Therefore, the mapping depends on the SIP server role that OSA SCS plays in a SIP session. For example, if SIP server in OSA SCS acts as a proxy server then the 2-party call has only one dialog in SIP (between the 2 UAs), while OSA MPCCS expects 2 legs (one from the calling party to OSA SCS and another from OSA SCS to the called party). Where an application demands full leg control in SIP the SIP server in OSA SCS should always act as UA (UA or B2BUA) or 3rd party controller . Only the latter modes of operation in SCS realises a direct 1:1 correlation between SIP dialog and OSA SCS MPCCS call leg.

4.2.1 OSA Call and SIP Dialogue Correlation Tables

Table 4-1: Parameter Correlation Proxy Mode, 2-party call

<table>
<tr><th>SIP</th><th>Headers</th><th></th><th>OSA API</th><th>Leg</th><th>CALL</th></tr>
<tr><td rowspan="4">SIP Dialog #1</td><td>call-ID(1)</td><td></td><td></td><td></td><td rowspan="4">callSessionID(1),

MPCCS
Call Object</td></tr>
<tr><td>local tag in From header(1)</td><td></td><td></td><td>callLegSessionID(1),

MPCCS
Originating Call Leg (1) object</td></tr>
<tr><td>remote tag in To header(1)</td><td></td><td></td><td rowspan="2">callLegSessionID(2),

MPCCS
Terminating Call Leg (2) object</td></tr>
<tr><td>Request-URI(1)</td><td></td><td>targetAddress(1)</td></tr>
<tr><td colspan="6">NOTE 1: The SIP server in OSA SCS is here acting as a stateful Proxy server. However, forking is NOT supported by current OSA API.
NOTE 2: The MPCCS callSessionID is assigned by the SCS and represents a correlation to the SIP call-id in the SIP INVITE request message. There should be no direct mapping as it would contradict SIP operation principles, i.e. the generation of a SIP call-ID for a particular invitation is the task of the inviting UA and the creation of a unique callSeesionID for an OSA application is the task of the SCS.
NOTE 3: The Call-ID identifies the call in the network. It is a global unique identifier.
The Request-URI is a SIP URL that indicates the user or service to which the request is being addresses and is used for routeing purpose.
The correlation shown corresponds to the case of an INVITE initial invitation from caller.</td></tr>
</table>

Table 4-2: Parameter Correlation B2BUA Mode, 2-party call

SIP	Headers		OSA API	Leg	CALL
SIP Dialog #1	**call-ID(1)**				callSessionID(1), MPCCS Call Object
	local tag in From header(1)			CallLegSessionID(1) MPCCS Originating Call Leg (1) Object	
	remote tag in To header(1)				
	Request-URI(1)		targetAddress(1)		
SIP Dialog #2	**call-ID(2)**				
	local tag in From header(1)				
	remote tag in To header(1)			CallLegSessionID(2), MPCCS Terminating Call Leg (2) object	
	Request-URI(1)		targetAddress(1/2) - may be changed by application.		
NOTE 1:	The B2BUA mode is comprised in the OSA SCS SIP server by two User Agents, acting as a User Agent Originating and a User Agent Terminating. It is a difficult implementation in SIP to shift from proxy mode into B2BUA mode and it is not possible in SIP to shift from B2BUA mode to proxy mode. Therefore where an application demands this mode of operation it has to be secured that it is established already at invitation request (INVITE). Notice: It is possible that only the call_ID(2) will be changed for the new SIP dialog #2 compared to SIP dialog #1as the incoming INVITE is "proxied". If a call forwarding application is invoked the targetAddress may be changed for routeing to the desired destination (Request URI).				
NOTE 2:	The MPCCS callSessionID is assigned by the SCS and represents a correlation to the SIP call-id in the SIP INVITE request message. There should be no direct mapping as it would contradict SIP operation principles, i.e. the generation of a SIP call-ID for a particular invitation is the task of the inviting UA and the creation of a unique callSeesionID for an OSA application is the task of the SCS.				
NOTE 3:	The Call-ID identifies the call in the network. It is a global unique identifier. The To header field contains the information regarding the endpoint who will receive the SIP request, e.g. INVITE or BYE message. The From header field represents the originator of the SIP request (e.g. the controller OSA SCS for SIP dialog #2). The Request-URI is a SIP URL that indicates the user or service to which the request is being addresses and is used for routeing purpose. The correlation shown corresponds to the case an INVITE initial invitation.				

Table 4-3: Parameter Correlation Originating UA Mode, 1-party call

SIP	Headers		OSA API	Leg	CALL
SIP Dialog #1	**call-ID(1)**				callSessionID(1), MPCCS Call Object
	local tag in From header(1)		value provided by OSA SCS)		
	remote tag in To header(1)			CallLegSessionID(1) MPCCS Terminating Call Leg (2) object	
	Request-URI(1)		targetAddress(1)		
NOTE 1:	The SIP server in OSA SCS is here acting as an User Agent Originating. The MPCCS callSessionID is assigned by the SCS and represents a correlation to the SIP call-id applied in the SIP dialogue. There should be no direct mapping as it would contradict SIP operation principles, i.e. the generation of a SIP call-ID for a particular invitation is the task of the inviting UA and the creation of a unique callSessionID for an OSA application is the task of the SCS.				
NOTE 2:	The Call-ID identifies the call in the network. It is a global unique identifier. The To header field contains the information regarding the endpoint who will receive the SIP request, e.g. INVITE or BYE message. The From header field represents the originator of the SIP request (e.g. the controller OSA SCS). The Request-URI is a SIP URL that indicates the user or service to which the request is being addresses and is used for routeing purpose. The correlation shown corresponds to the case of an INVITE initial invitation.				

Table 4-4: Parameter Correlation Terminating UA / Redirection Mode, 1-party call

SIP	Headers		OSA API	Leg	CALL
SIP Dialog #1	**call-ID(1)**				callSessionID(1), MPCCS Call Object
	local tag in From header(1)			CallLegSessionID(1). MPCCS Originating Call Leg (1) object	
	remote tag in To header(1)		(value provided by OSA SCS)		
	Request-URI(1)		address(1)		

NOTE 1: The SIP server in OSA SCS is acting as a User Agent Terminating.
The OSA MPCCS API allows the application to instruct the return of a final SIP response (2xx, 3xx, 4xx, 5xx, 6xx) to a received SIP request (INVITE) .Note1: The MPCCS callSessionID is assigned by the SCS and represents a correlation to the SIP call-id applied in the SIP dialogue. There should be no direct mapping as it would contradict SIP operation principles, i.e. the generation of a SIP call-ID for a particular invitation is the task of the inviting UA and the creation of a unique callSeesionID for an OSA application is the task of the SCS.

NOTE 2: The Call-ID identifies the call in the network. It is a global unique identifier.
The To header field contains the information regarding the endpoint who will receive the SIP request, e.g. INVITE or BYE message. The From header field represents the originator of the SIP request. The Request-URI is a SIP URL that indicates the user or service to which the request is being addresses and is used for routeing purpose.
The correlation shown corresponds to the case of an INVITE initial invitation.

Table 4-5: Parameter Correlation 3rd party controller Mode, 2-party call

SIP	Headers		OSA API Parameters	Leg	CALL
SIP Dialog #1	**call-ID(1)**		-		**callSessionID(1)** See Note1. MPCCS Call Object
	local tag in From header(1)		(provided by OSA SCS may be used)		
	Remote tag in To header(1)			**callLegSessionID(1)** MPCCS Terminating Call Leg (1) object.	
	Request-URI(1)		**targetAddress(1)**		
SIP Dialog #2	**call-ID(2)**		-		
	local tag in From header(1)		(value provided by OSA SCS may be used)		
	To header(2)			**callLegSessionID(2),** MPCCS Terminating Call Leg (2) object	
	Request-URI(2)		targetAddress (2)		

NOTE 1: The 3.rd party controller mode is comprised in the OSA SCS SIP server by two or more User Agents , in this example by two User Agents Originating.
Not possible in SIP to shift from proxy mode into 3rd party controller mode. Therefore where an application demands this mode of operation it has to be secured that it is established already at invitation request (INVITE).

NOTE 2: Same callSessionID(1) used by the application in the creation of both the OSA Call Leg objects as both legs are to be part of the same call.

NOTE 3: The Call-ID identifies the call in the network. It is a global unique identifier.
The To header field contains the information regarding the endpoint who will receive the SIP request, e.g. INVITE or BYE message. The From header field represents the originator of the SIP request. The Request-URI is a SIP URL that indicates the user or service to which the request is being addresses and is used for routeing purpose.
The correlation shown corresponds to the case of an INVITE initial invitation.

5 Multi Party Call Control Flows

NOTE: The Call Flows in the following are to be regarded as example flows. They are merely intended to illustrate theSIP mapping from/to OSA APIs and do not necessary provide complete SIP call/session flows. More detailed SIP call flows are defined in [13].
Additional information including the different SIP server modes of operation for OSA SCS in relation to MPCCS mapping is found in Annex A "Introduction to API Mapping for OSA MPCCS".

5.1 Call Manager Service Interface

The call manager interface class provides the management functions to the multi-party call Service Capability Features. The application programmer can use this interface to create call objects and to enable or disable call-related event notifications.

5.1.1 CreateCall

createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier

This method is used to create a new Call object in the SCS.

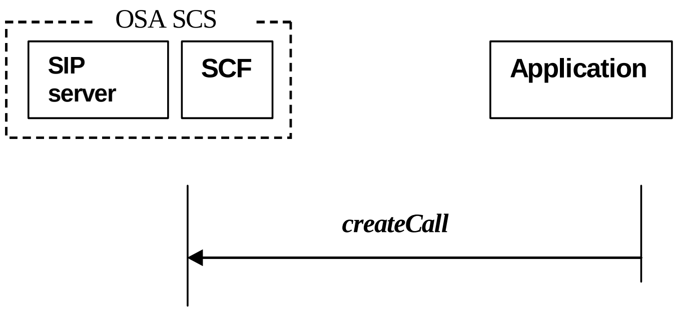

Figure 5-1: Call flow for *createCall()*

Table 5-1: Normal operation

SIP Server Mode for the OSA SCS:	**UA mode**
Pre-conditions:	**An agreement is established between the network operator and the service provider to enable the application to create call object.**
1	A new Multi-party Call object is created in the SCS and the application gets a reference to the call object.

Table 5-2: Parameter Mapping

From: createCall	To: SIP	Remark
appCall (IpAppMultiPartyCallRef)	N/A	No mapping.
Returns: TpMultiPartyCallIdentifier: - CallReference **(IpMultiPartyCallRef)** - CallSessionID **(TpSessionID)**	N/A	Not mapped. However, the call Session ID returned in this method will later on be correlated to the applied SIP call-Id

5.1.2 CreateNotification

createNotification (appCallControlManager : in IpAppMultiPartyCallControlManagerRef, notificationRequest: in TpCallNotificationRequest) : TpAssignmentID

This method is used to enable call notifications so that events can be sent to the application. The interface between DB (HSS) and OSA SCS is Sh interface, for detail see 3GPP TS 29.328 [15].

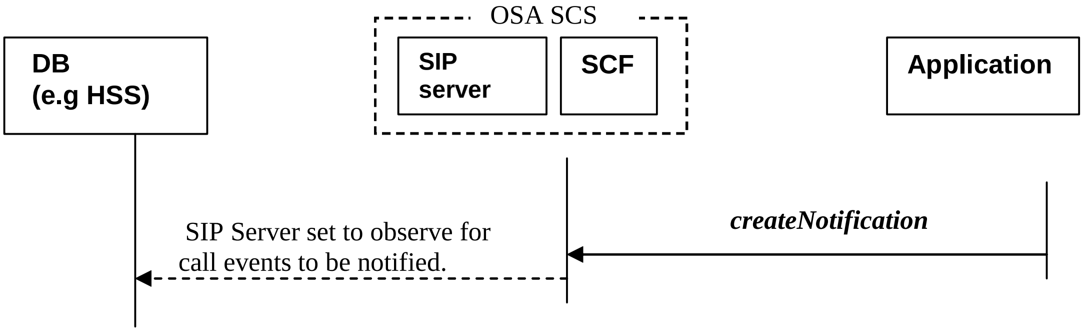

Figure 5-2: Call flow for *createNotification()*

Table 5-3: Normal Operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd Party controller.** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the event notification to be enabled**
1	The application invokes the ***createNotification*** method
2	The SCS requests the controlled SIP server to observe for certain SIP call events to be notified to the application. Initial filtering information will be uploaded to the DB (Data Base e.g. HSS) and from here to controlled entity (e.g. S-CSCF), e.g. when the user gets registered.
NOTE: The ***createNotification*** represents the first step an application has to do to get initial notifications of calls happening in the network. When such an event happens, the application will be informed by ***reportNotification*** However, ***createNotification()*** is not applicable if the call is set-up from the network by the application.	

Table 5-4: Parameter Mapping

From: createNotification	To: SIP	Remark
appCallControlManager (IpAppMultiPartyCallControlManagerRef)	N/A	If set it specifies a reference to the application interface, which is used for call-backs.
notificationRequest (TpCallNotificationRequest) :	See table 6-15: **TpCallNotificationRequest** for the mapping from SIP.	Specifies the event specific criteria used by the application to define the event required. Not mapped to SIP. However, the parameter has to be verified for SIP validity of parameter values.
Returns: **TpAssignmemtID**	N/A	Returns assignmentID to application, which specifies the ID assigned by the multi party call control manager interface for this newly enabled event notification.
NOTE: No direct mapping to SIP. However, the SIP server responsible for event filtering (e.g. S-CSCF) is to monitor for SIP events requested to be notified to the application if encountered and conditions (filter criteria) for reporting are fulfilled.		

5.1.3 changeNotification

changeNotification (assignmentID : in TpAssignmentID, notificationRequest : in TpCallNotificationRequest) : void

This method is used by the application to change the call notifications previously set by ***createNotification*** .

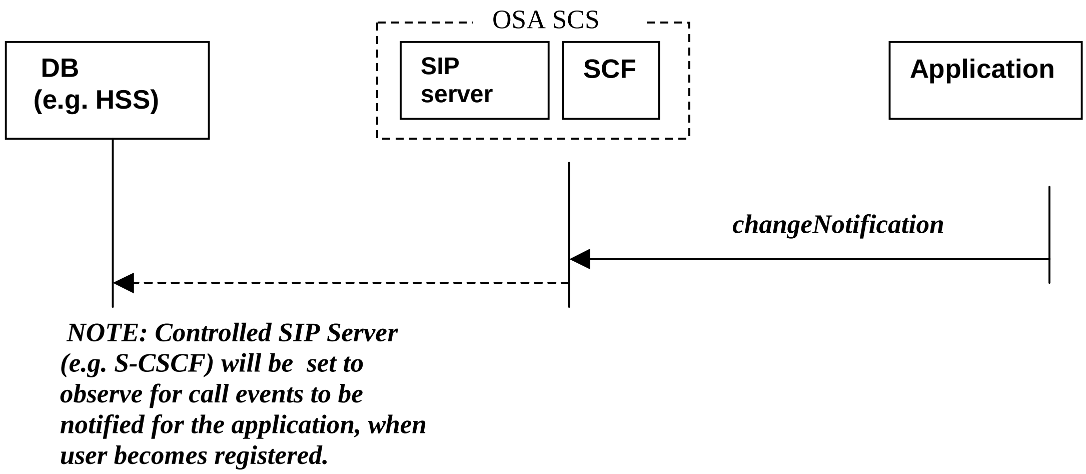

Figure 5-3: Call flow for *changeNotification()*

Table 5-5: Normal Operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd Party controller.** **Note: The applicable mode will depend on the behaviour of the application on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the event notification to be enabled. Notifications have been enabled by the application**
1	The application invokes the ***changeNotification*** method
2	The SCS requests a change in the set of initial notifications, i.e. initial filtering information is changed. Note: Updated initial filtering information will be uploaded to the DB (Data Base e.g. HSS) and from here to the controlled entity (e.g. S-CSCF), e.g. when the user gets registered.

Table 5-6: Parameter mapping

From: changeNotification	To: SIP	Remark
assignmentID (TpAssignmentID)	N/A	Specifies the ID assigned by the multi party call control manager interface for the event notification.
notificationRequest (TpCallNotificationRequest) :	See table 6-15: **TpCallNotificationRequest** for the mapping from SIP.	Not mapped directly to SIP. However, the parameter has to be verified for SIP validity of parameter values.
NOTE: No direct mapping to SIP. However, the SIP server responsible for event filtering (e.g. S-CSCF) is to monitor for SIP events requested to be notified to the application if encountered and conditions (filter criteria) for reporting are fulfilled.		

5.1.4 destroyNotification

destroyNotification (assignmentID : in TpAssignmentID) : void

This method is used by the application to disable call notifications.

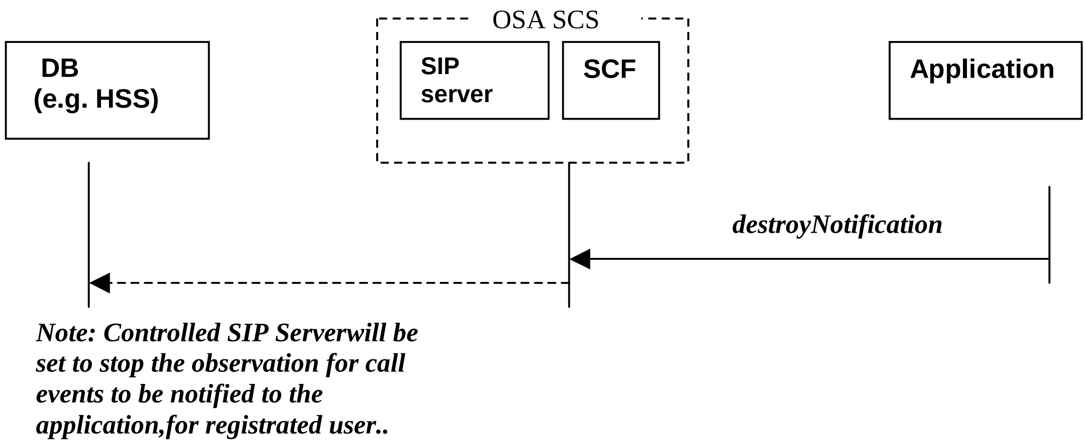

Figure 5-4: Call flow for *destroyNotification()*

Table 5-7: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd Party controller.** **Note: The applicable mode will depend on the behaviour of the application on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the event notification to be disabled.**
1	The application invokes the ***destroyNotification*** method
2	The SCS requests to de-activate the active call notification.
NOTE: Destroyed notifications (initial filtering) information will be uploaded to the DB (Data Base e.g. HSS) and from here to the controlled entity (e.g. S-CSCF), if the user has been registered.	

Table 5-8: Parameter Mapping

From: destroyNotification	To: SIP	Remark
assignmentID (TpAssignmentID)	N/A	Specifies the ID assigned by the multi party call control manager interface for the event notification.

5.1.5 getNotification

getNotification () : TpNotificationRequestedSet

This method is used by the application to query the event criteria set previously using ***createNotification*** *and possibly* ***changeNotification***.

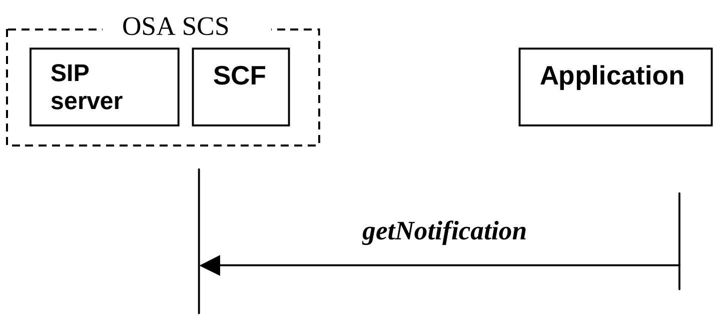

Figure 5-5: Call flow for *getNotification()*

Table 5-9: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller** **Note: The applicable mode will depend on the behaviour of the application on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the event notification. Notifications have been enabled by the application.**
1	The application invokes the ***getNotification*** method.
2	The OSA SCS returns the criteria as set for event notification.

Table 5-10: Parameter mapping

From: getNotification	To: SIP	Remark
Returns: TpNotificationRequestedSet: A set of **TpNotificationRequested:**	-	No SIP mapping.
- AppCallNotificationRequest (TpCallNotificationRequest)	N/A	Returns information as previously set in ***createNotification*** and ***changeNotification***.
- **AssignmentID** (TpInt32)	N/A	
NOTE: The set of all previously requested notification events are returned. No mapping to SIP. The method ***getNotification*** contains no parameter – only a return parameter exists.		

5.1.6 setCallLoadControl

setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange) : TpAssignmentID

This method is used to impose or remove load control on calls made to a specific address within the call control service.

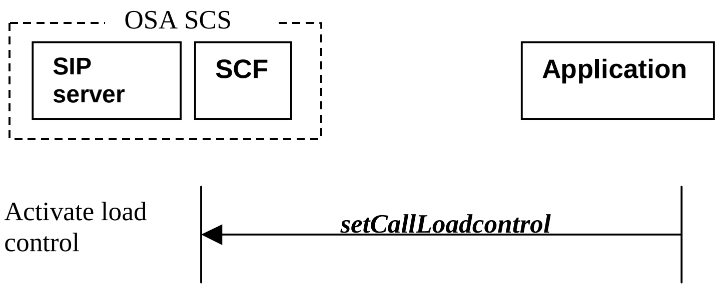

Figure 5-6: Flow for *setCallLoadControl()*

Table 5-11: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller.** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the set call load control.**
1	The application invokes the ***setCallLoadControl*** method to remove or set load control on calls made to a specific address or address range.
2	The SCS requests the SIP server to activate or remove call load control

Table 5-12: Parameter Mapping

From: setCallLoadControl	To: SIP	Remark
duration (TpDuration)	N/A	-
mechanism (TpCallLoadControlMechanism)	N/A	Specifies the applied load control mechanism and defines the call admission rate (e.g. allow one call per interval).
treatment (TpCallTreatment) TpCallTreatment sequence of: - TpCallTreatmentType, - TpReleaseCause	See Table 6-16 **TpCallTreatment Type** **and Table 6-18 TpReleaseCause** for the mapping to SIP	Specifies how to treat (e.g. deny) new invitations if overload prevails.
addressRange (TpAddressRange)	See Table 6-3: **TpAddressRange** for the "mapping" from SIP.	Specifies the address or address range to which overload control should be applied or removed. Not mapped directly but has to be verified for application with SIP URL.

5.2 Call Manager Application Interface

5.2.1 managerInterrupted

managerInterrupted () : void

This method is used to indicate to the application that all event notifications and method invocations have been temporarily interrupted, for example due to network resources unavailable.

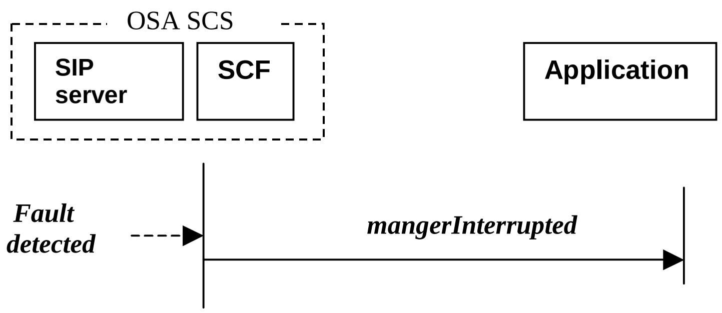

Figure 5-7: Call flow for *managerInterrupted()*

Table 5-13: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the call notification. Call notifications have been enabled using the *createNotification* method on the Call Manager interface.**
1	The SCS has detected, or has been informed of a fault which prevents further events from being notified to the application.
2	The SCS invokes the ***managerInterrupted*** method.

Table 5-14: Parameter Mapping

From: *managerInterrupted*	To: SIP	Remark
-	N/A	No parameters in this method.

5.2.2 managerResumed

managerResumed () : void

This method is used to indicate to the application that all event notifications are possible and method invocations are enabled after having previously been interrupted.

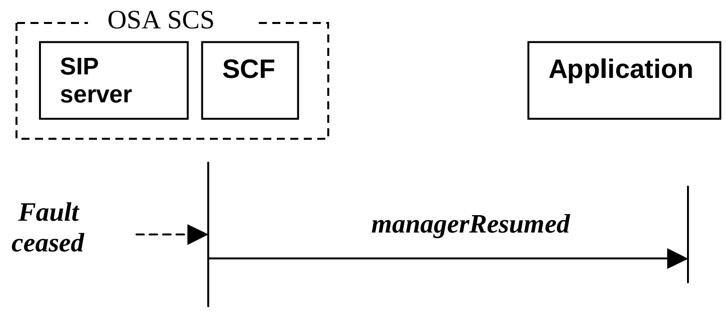

Figure 5-8 Call Flow for *managerResumed()*

Table 5-15: Normal Operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**An agreement is established between the network operator and the service provider for the call notification. Call notifications have been interrupted and *managerInterrupted* method has been invoked.**
1	The SCS detects that call notifications are again possible.
2	The SCS invokes the ***managerResumed*** method.

Table 5-16: Parameter Mapping

From: managerInterrupted	To: SIP	Remark
-	N/A	No parameters in the method.

5.2.3 reportNotification

reportNotification (callReference : in TpMultiPartyCallIdentifier, callLegReferenceSet : in TpCallLegIdentifierSet, notificationInfo : in TpCallNotificationInfo, assignmentID : in TpAssignmentID) : TpAppMultiPartyCallBack

This method is used to notify the application of the arrival of a call-related event. It is sent in response to the ***createNotification()*** method.

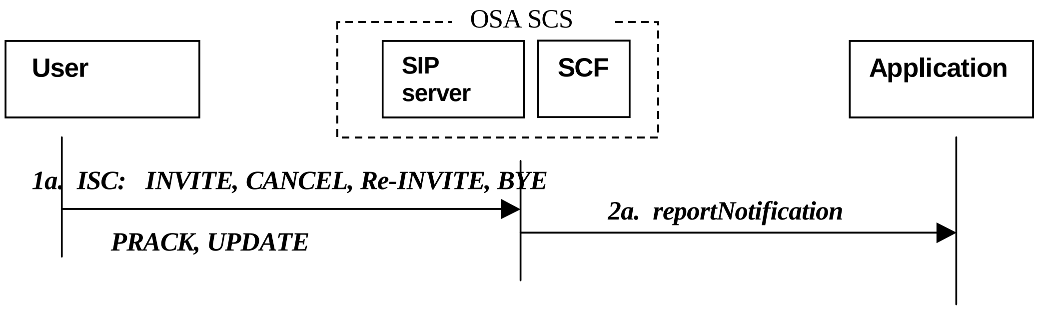

Figure 5-9: Call flow for *reportNotification*, triggered by SIP requests

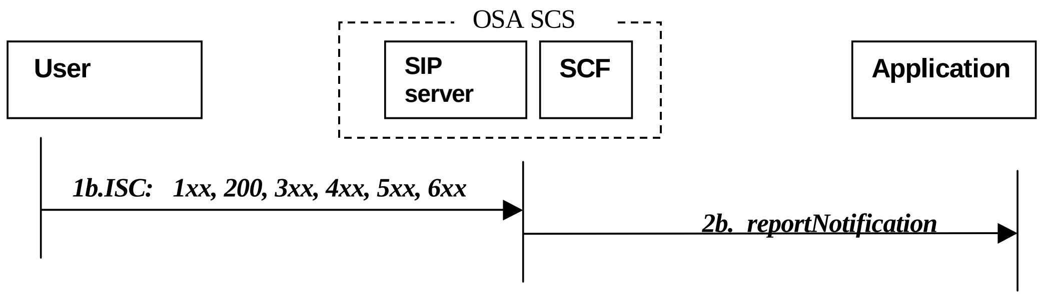

Figure 5-10: Call flow for *reportNotification*, triggered by SIP Reponses

Table 5-17: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA,3rd Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**Call notifications have been enabled using the *createNotification* method on the Call Manager interface.**
1	A call arrives from a call party or terminates to a call party or a call party decides to issue a mid-call event or terminate the involvement in an established call. This request is detected by the SIP server and the criteria for an initial notification to be reported is checked.
2	When the criteria for an initial notification is met, the SCS identifies the application responsible for handling the call and invokes the ***reportNotification*** method.

Table 5-18: Parameter Mapping

To: reportNotification	From: SIP	Remark
callReference (TpMultiPartiCallIdentifier) TpMultiPartyCallIdentifier: **- CallReference (**IpMultiPartyCallRef) **- CallSessionID** (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	The SCS will create a new call object and associated call leg object and pass them to the application. A correlation between SIP call-ID and call session ID is created.
callLegReferenceSet (TpCallIdentifierSet). A set of **TpCallIdentifier:**	**-**	
- **CallLegreference** (IpCallLegRef)	**N/A**	This element specifies the interface for the Call Leg object.
- CallLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	This element specifies the call leg session ID. No direct mapping to SIP – but a correlation is created.
notificationInfo (TpCallNotificationInfo):	-	
-TpCallNotificationReportScope	See Table 6-14 : **TpCallNotificationReport Scope**	
- CallAppInfo (TpCallAppInfoSet) Note: A set of **TpCallAppInfo**	See Table 6-4: **TpCallAppInfo**	
- CallEventInfo (TpCallEventInfo)	See Table 6-7: **TpCallEventInfo**	
assignmentID (TpAssignmentID)	N/A See note:	Specifies the assignment id which was returned by the createNotification() method. The application can use assignment id to associate events with specific criteria and to act accordingly.
NOTE: Indeed the assignmentiD does not involve SIP mapping, it could be stored in the OSA SCS. .		

5.2.4 callAborted

callAborted (callReference : in TpSessionID) : void

This method is used to indicate to the application that the call object has aborted or terminated abnormally. No further communication will be possible between the call and the application.

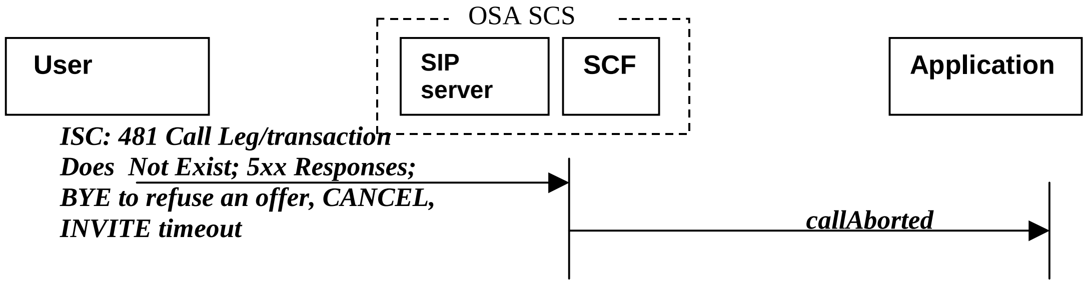

Figure 5-11: Call flow for *callAborted()*

Table 5-19: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**The SCS detect a failure in its communication with the SIP server**
1	The SCS, invokes the ***callAborted*** method. Since the SIP server reflects the call running in the network, the call could also have been aborted in the network.

Table 5-20: Parameter Mapping

From: callAborted	To: SIP	Remark
callReference (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	Specifies the sessionID of the call that has aborted or terminated abnormally. No direct mapping to SIP – but a correlation is created.

5.2.5 callOverloadEncountered

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

This method is used to indicate that the network has detected overload and may have automatically imposed load control on calls requested to a particular address range or calls made to a particular destination within the call control service.

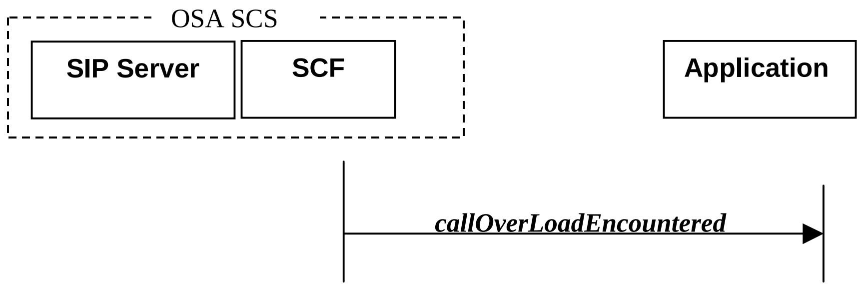

Figure 5-12: Call flow for *callOverLoadEncountered()*

Table 5-21: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**Call overload control have been enabled using the *setCallOverloadControl* method on the Call Manager interface.**
1	The SCS detect a call overload situation in its communication with the SIP server of the OSA SCS.
2	The SCS, invokes the ***callOverLoadEncountered*** method. The call running in the network may continue or not depending on the requested treatment at overload (defined by ***setCallOverloadControl*** method received previously).

Table 5-22: Parameter Mapping

From: callOverloadEncountered	To: SIP	Remark
assignmentID (TpAssignmentID)	**N/A.**	Specifies the assignmentID corresponding to the associated setCallLoadControl method. This implies the address or address range within which the overload has been encountered (the SIP URL(s)).

5.2.6 callOverloadCeased

callOverloadCeased (assignmentID : in TpAssignmentID) : void

This method is used to indicate that the network has detected that the overload has ceased and has automatically removed any load controls on calls requested to a particular address range or calls made to a particular destination within the call control service.

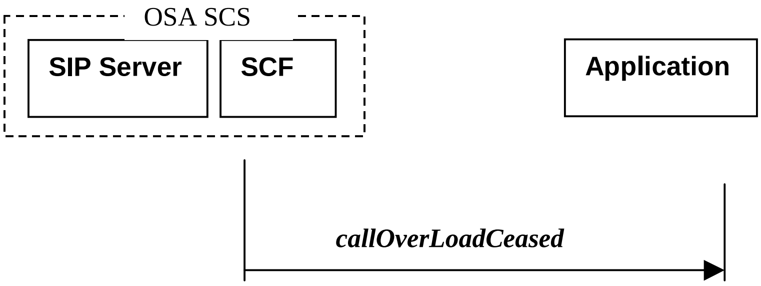

Figure 5-13: Call flow for *callOverLoadCeased()*

Table 5-23: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller.** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**The network has detected overload and may have automatically imposed load control on calls requested to a particular address or address range.**
1	The SCS detect that an overload situation has ceased in its communication with the SIP server
2	The SCS, invokes the ***callOverLoadCeased*** method.

Table 5-24: Parameter Mapping

From: callOverloadEncountered	To: SIP	Remark
assignmentID (TpAssignmentID)	**N/A.**	Specifies the assignmentID corresponding to the associated setCallLoadControl method. This implies the address or address range within which cease of overload has been encountered (the SIP URL(s)). No mapping to SIP – but an association is created, see mapping for ***setCallOverloadControl.***

5.3 Multi-Party Call Service Interface

The multi-party call interface class represents the interface to the multi-party call Service Capability Feature. It provides a structure to allow simple and complex call behaviour.

5.3.1 GetCallLegs

getCallLegs (callSessionID : in TpSessionID) : TpCallLegIdentifierSet

This method is used to obtain references to the current Call Leg objects, associated to the Multi-party call object.

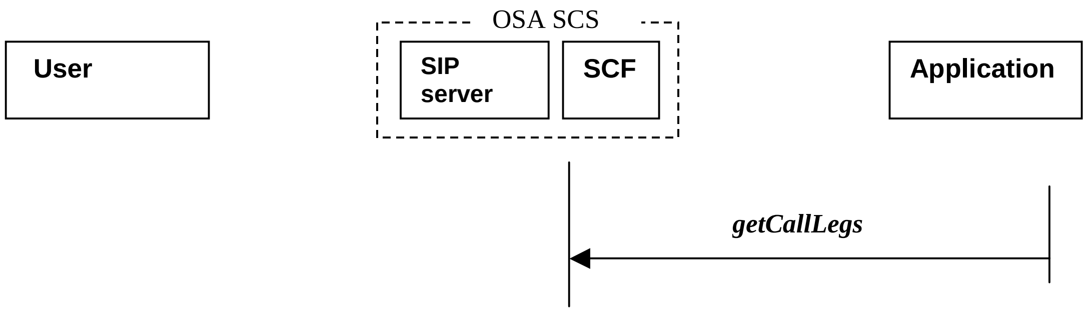

Figure 5-14: Call flow for *getCallLegs()*

Table 5-25: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, Redirect, UA, B2BUA, 3rd. Party controller** **Note: The applicable mode will depend on the behaviour of the application invoked on the call.**
Pre-conditions:	**The application has a reference to a Multi-party Call object.**
1	The application invokes the ***getCallLegs*** method
2	The SCS returns information about the involved call leg objects

Table 5-26: Parameter mapping

From: callOverloadEncountered	To: SIP	Remark
callSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	Specifies the call session ID of the call. No direct mapping to SIP – but a correlation is created.

5.3.2 createCallLeg

createCallLeg (callSessionID : in TpSessionID, appCallLeg : in IpAppCallLegRef) : TpCallLegIdentifier

This method is used to create a new CallLeg object in the SCS.

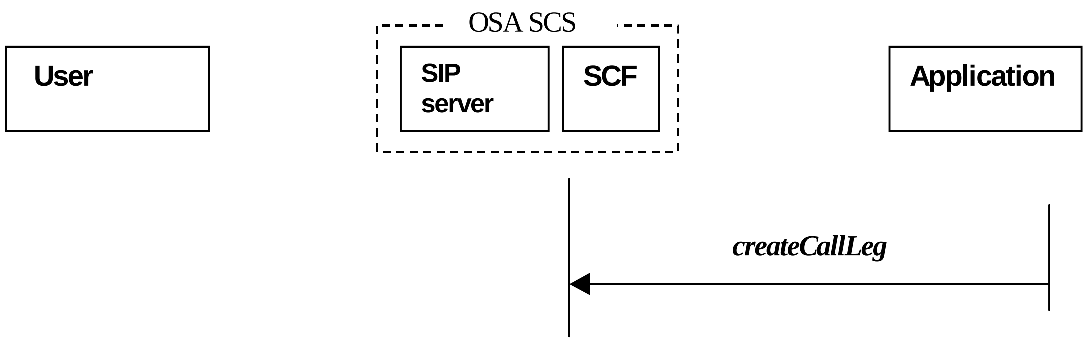

Figure 5-15: Call flow for *createCallLeg()*

Table 5-27: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any, except Redirect).**
Pre-conditions:	**The application has a reference to a Multi-party Call object.**
1	The application invokes the ***createCallLeg*** method
2	The SCS creates the requested call leg object

Table 5-28: Parameter mapping

From: callOverloadEncountered	To: SIP	Remark
callSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	Specifies the call session ID of the call. No direct mapping to SIP – but a correlation is created.
appCallLeg (IpAppCallLegRef	N/A	Specifies the application interface for call-backs from the call leg created
Returns: **TpCallLegIdentifier:** **- CallLegReference (**IpCallLegRef) **- CallLegSessionID** (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	The SCS will create a new call leg object to be associated with the existing call object and pass it to the application.
NOTE: The correlation to SIP will be created when set-up of a connection associated with the created call leg occurs.		

5.3.3 createAndRouteCallLegReq

createAndRouteCallLegReq (callSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet, targetAddress : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, appLegInterface : in IpAppCallLegRef) : TpCallLegIdentifier

This method is an asynchronous method used to request the creation of a new Call Leg and the set-up of a connection to the indicated address.

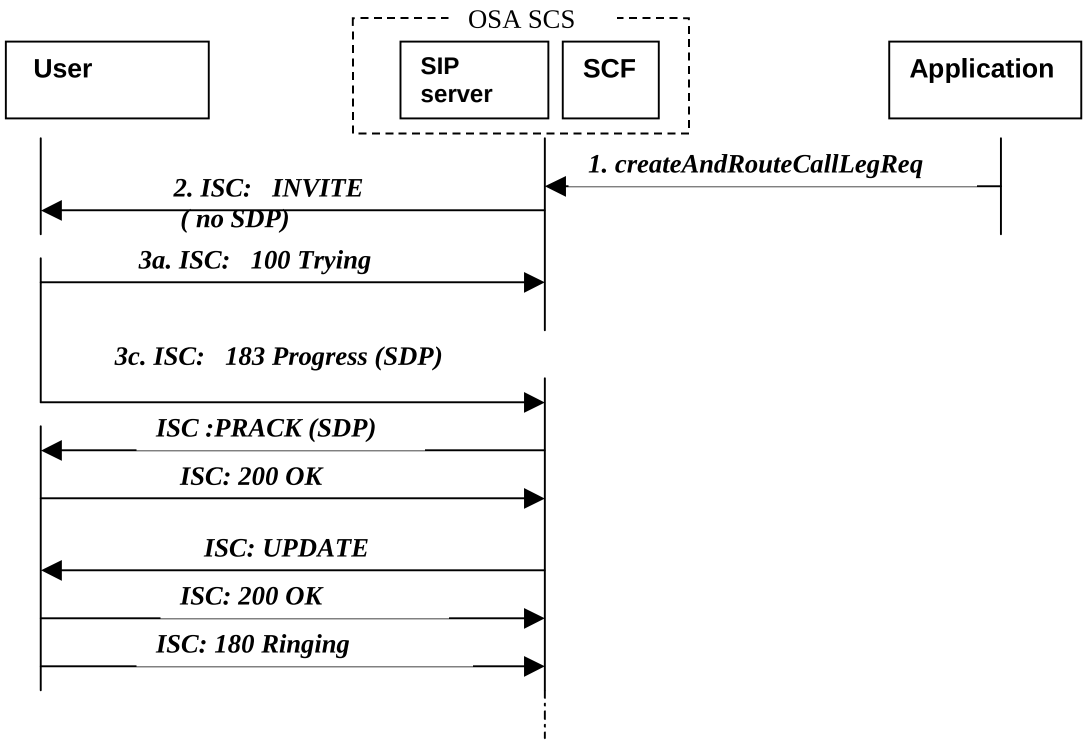

Figure 5-16: Call flow for *createAndRouteCallLegReq()*, OSA SCS acting as UA Client

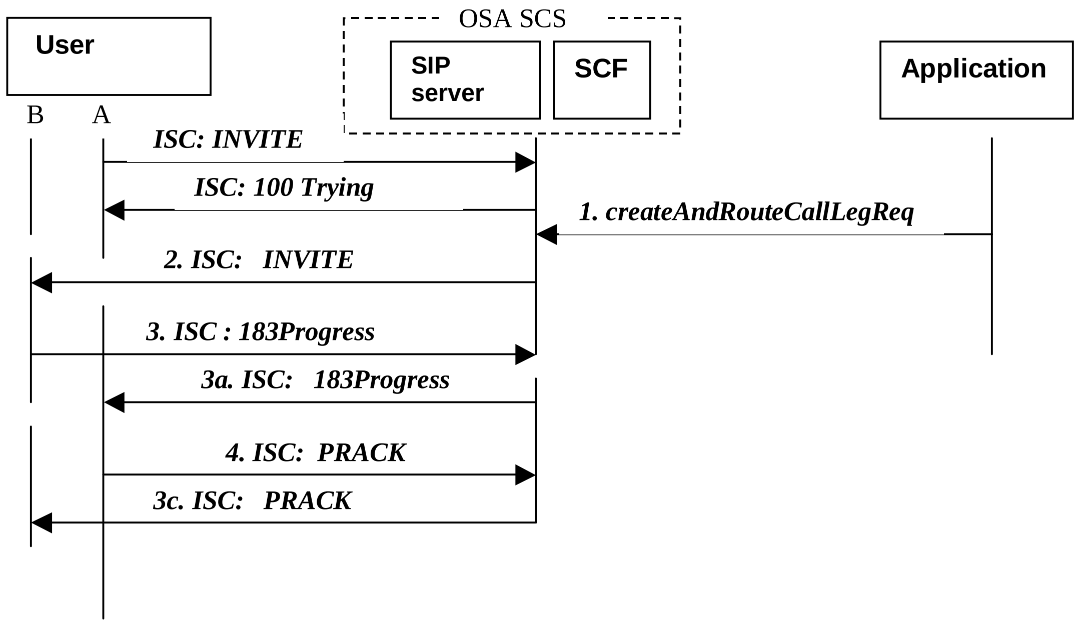

Figure 5-17: Call flow for *createAndRouteCallLegReq()*, OSA SCS acting as Proxy server

Table 5-29: Normal operation, case a: UA mode

SIP Server Mode for the OSA SCS:	**UA (or 3rd party controller, B2BUA).**
Pre-conditions:	**The application has a reference to a Multi-party Call object.**
1	The application invokes the ***createAndRouteCallLegReq*** method. The SCS creates an call Leg object and instructs the SIP server of the OSA SCS to generates a SIP INVITE message.
2	The SIP server acting in a UA mode sends the SIP INVITE to the corresponding party. Note: It may happen that the destination address leads to the generation of more than one INVITE being sent by the SIP server (Forking).
3	The SIP server acting as UA acknowledge the incoming **SIP response message**.
NOTE 1: The application has no control of the SIP server forking functionality. Assuming the UA ("surrogate UAC") of the OSA SCS does not posses any media resource, the INVITE is sent with "no SDP". This results in a SIP dialog with no media (e.g. no RTP stream) stream set-up, i.e. a plain session control dialog created by the application. The possible handling of media by "UA" within the OSA SCS for application initiated calls is outside the scope of standardisation. NOTE 2: See also Annex B for supplementary information and flow examples (B2- B5) (CreateAndRouteCallLegReq may hereby be viewed as a concatenation the methods createCallLeg, eventReportReq and routeReq).	

Table 5-30: Parameter mapping, UA mode

From: *createAndRouteCallLegReq*	**To: SIP** INVITE	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" for Originating UA mode. Table 4-2 to 4-5.	No direct mapping, merely a correlation is created.
eventsRequested (TpCallEventRequestSet) Note: A set of TpCallEventRequest	See Table 6-8: TpCallEventRequest for mapping to SIP.	Start observation in SIP server for occurrence of requested events to be notified to the application.
targetAddress (TpAddress)	SIP URL in the TO header and Request-URI See Table 6-2: TpAddress mapping to SIP.	
originatingAddress (TpAddress)	SIP URL in the From header. See Table 6-2: TpAddress mapping to SIP.	The originating address may e.g. be the application server SIP address (third party call set up) or the SCS server when the the SCS is the endpoint (UAC) which initiates the INVITE. If originatingAddress not present a default value could be provided by the OSA SCS.
appInfo (TpCallAppInfoSet) Note: A set of TpCallAppInfo	See Table 6-4: TpCallAppInfo for mapping to SIP.	
appLegtInterface (IpAppCallLegRef)	N/A	Defines a reference to data type IPCallLeg
Returns: **TpCallLegIdentifier:** **- CallLegReference** (IpCallLegRef) **- CallLegSessionID** (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-2 to 4-5.	A correlation to SIP is created. The SCS will create a new call leg object to be associated with the existing call object and pass it to the application. Note: The correlation to SIP is created when set-up of a connection associated with the created call leg occurs..
NOTE: See also Annex B and Annex C .		

Table 5-31: Normal operation, case b: Proxy mode

SIP Server Mode for the OSA SCS:	**Proxy.**
Pre-conditions:	**The application has a reference to a Multi-party Call object.**
1	The application invokes the ***createAndRouteCallLegReq*** method. The SCS creates an call Leg object, and forwards the received SIP INVITE message to the indicated target address.
2	The SIP server forwards the SIP INVITE to the corresponding party. Note: It may happen that the destination address leads to the generation of more than one INVITE being sent by the SIP server (Forking).
3	The SIP server forwards the incoming **SIP response message** to the SCS.
NOTE: The application has no control of the SIP server forking functionality.	

Table 5-32: Parameter mapping, Proxy mode

From: *createAndRouteCallLegReq*	**To: SIP** INVITE	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" for Proxy mode. Table 4-1.	No direct mapping of CallSessionID onto SIP Call-ID to ensure the SIP Call-ID uniqueness, merely a correlation is needed. A SIP call ID must be unique and not be reused for later calls. Acting as a UA (or B2BUA) a new call_ID is created for the new originating SIP leg for which a correlation with callSessionID is created.
eventsRequested (TpCallEventRequestSet) Note: A set of TpCallEventRequest	See Table 6-8: TpCallEventRequest for mapping to SIP	Start observation in SIP server of the OSA SCS for occurrence of requested events to be notified to the application.
targetAddress (TpAddress)	SIP URL in the Request URI header. See Table 6-2: TpAddress mapping to SIP.	If present, the targetAddress is used for routeing using Request-URI
originatingAddress (TpAddress)	N/A	FROM header containf the originator address (caller) of the invitation. This must not be changed.
appInfo (TpCallAppInfoSet) Note: A set of TpCallAppInfo	See Table 6-4: TpCallAppInfo for mapping to SIP.	
appLegtInterface (IpAppCallLegRef)	N/A	Defines a reference to data type IPCallLeg
Returns: **TpCallLegIdentifier:** **- CallLegReference** (IpCallLegRef) **- CallLegSessionID** (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-1.	A correlation to SIP is created. The SCS will create a new call leg object to be associated with the existing call object and pass it to the application. Note: The correlation to SIP is created when set-up of a connection associated with the created call leg occurs..
NOTE: See also Annex B and Annex C.		

5.3.4 release

release (callSessionID : in TpSessionID, cause : in TpReleaseCause) : void

This method used to request the release of the call and associated objects.

Remarks: If several legs are connected, this method will also release each of the call legs, i.e. the complete call is released. The flow example below indicates the release of a single user (call party), it is however applicable for the release of any user, i.e. BYE is to be sent for each user (SIP dialog) that take part in the call.

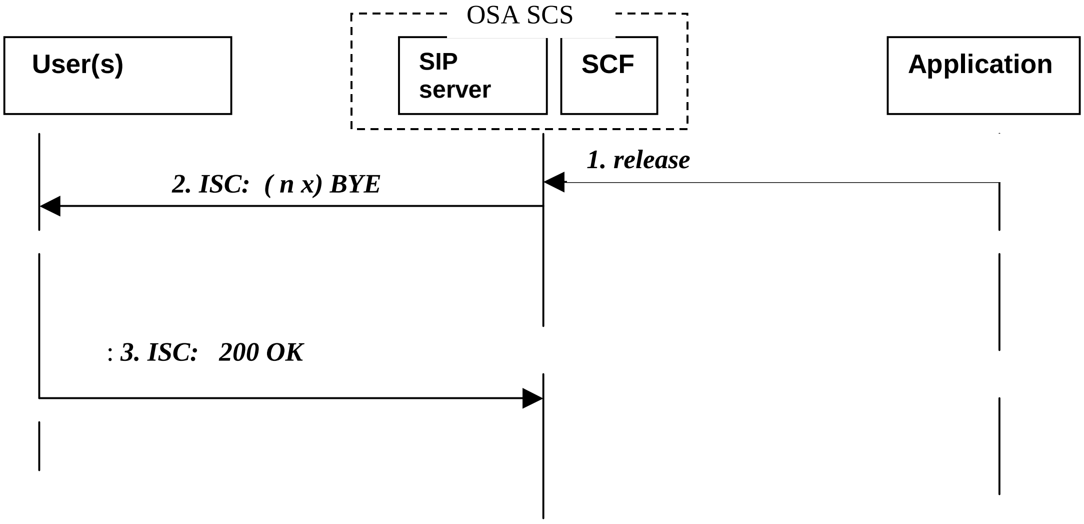

Figure 5-18: Call flow for *release*, acting as UA (incl. B2BUA, 3rd. Party Controller)

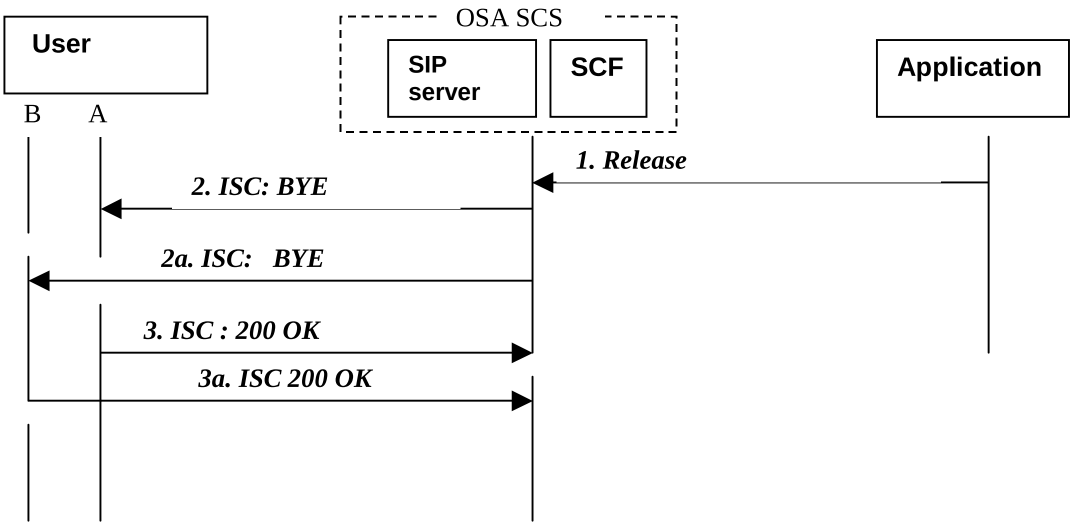

Figure 5-18a: Call flow for *release*, acting as proxy

Table 5-33: Normal operation, UA mode

SIP Server Mode for the OSA SCS:	**UA (or 3rd party controller, B2BUA).** **For call release from application, UA mode of operation is demanded.**
Pre-conditions:	Call is in progress. **The application has a reference to a Multi-party Call object.**
1	The application invokes the ***release*** method. For all legs associated to the call, the SCS will act as if a ***release()*** method was received for each present leg(s).
2	If the application has requested some reports at the end of the call (e.g., getInfoReq(), superviseReq()) these reports will be sent to the application
3	
NOTE 1: The SIP server of the SCS gateway is to be capable to issue the SIP BYE to release the call participant(s) on request from the application - and therefore it demands to play the role of a UA. NOTE 2: Release may be sent any time from the application e.g. resulting in creation of a SIP response (e.g. 4xx, 5xx) to an incoming INVITE request or the termination of an establishment session (BYE) or the cancellation of a pending request (CANCEL) after the application has issued an INVITE request.	

Table 5-33a: Normal operation, proxy mode

SIP Server Mode for the OSA SCS:	**Proxy**
Pre-conditions:	Call is in progress. **The application has a reference to a Multi-party Call object.**
1	The application invokes the ***release*** method. For all legs associated to the call, the SCS will act as if a ***release()*** method was received for each present leg(s).
2	If the application has requested some reports at the end of the call (e.g., getInfoReq(), superviseReq()) these reports will be sent to the application
3	
NOTE 1: The SIP server of the SCS gateway is to be capable to issue the SIP BYEs to multiple call participants on request from the application - and therefore it acts as a transparent B2BUA which remembers the sequence number of the requests sent by the call participants. NOTE 2: Release may be sent any time from the application e.g. resulting in creation of a SIP response (e.g. 4xx, 5xx) to an incoming INVITE request or the termination of an establishment session (BYE) or the cancellation of a pending request (CANCEL) after the application has issued an INVITE request.	

Table 5-34: Parameter mapping

From: *release*	**To: SIP** BYE, 4xx, 5xx, Cancel (if any pending INVITE requests from application)	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-2 to 4-5.	No direct mapping, merely a correlation is created.
cause (TpReleaseCause) :	See table 6-17: **TpReleaseCause** for mapping to SIP response codes	See also note below
NOTE: The release() method may be sent any time from the application e.g. resulting in a) creation of a SIP response (e.g. 4xx, 5xx) to an incoming INVITE request or b) the termination of an established session (BYE) or c) the cancellation of pending requests (CANCEL) when the application has issued an INVITE request.		

5.3.5 deassignCall

deassignCall (callSessionID : in TpSessionID) : void

This method is used to request that the relationship between the application and the call and associated objects be de-assigned. It leaves the call in progress, however, it purges the specified call object so that the application has no further control of call processing. If a call is de-assigned that has event reports or call information reports requested, then these reports will be disabled and any related information discarded.

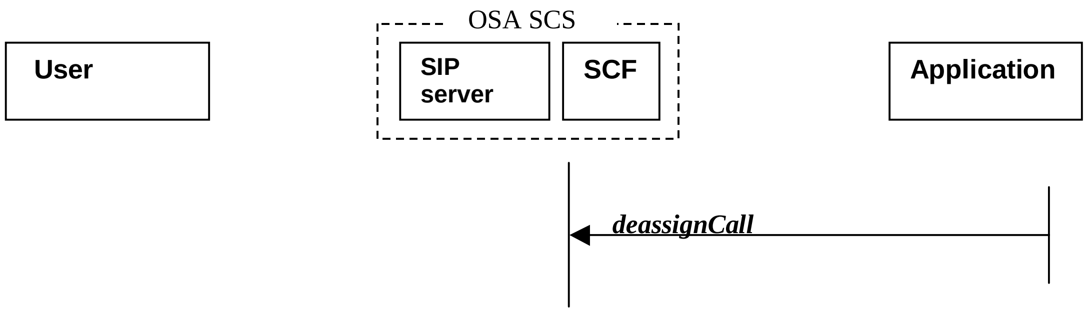

Figure 5-19: Call flow for *deassignCall()*

Table 5-35: Normal operation

SIP Server Mode for the OSA SCS:	Proxy, UA, B2BUA or 3rd party controller, Redirect
Pre-conditions:	**A relationship between the application and the call including associated objects exists.**
1	The application invokes the ***deassignCall*** method
2	The SCS terminates the relationship between the application and the call and its associated objects and notifies the SIP server of the OSA SCS.
3	The SIP server of the OSA SCS is to continue call processing autonomously, i.e. without any control from the application. Any possible interrupted call processing is to be resumed.
NOTE: If the application was the only one to control the session, the SIP server of the OSA SCS may remove itself from the route-request.	

Table 5-36: Parameter mapping

From: *release*	To: SIP	Remark
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	No direct mapping, merely a correlation is created.

5.3.6 getInfoReq

getInfoReq (callSessionID : in TpSessionID, callInfoRequested : in TpCallInfoType) : void

This method is an asynchronous method that requests information associated with the call to be provided at the appropriate time (for example, to calculate charging).

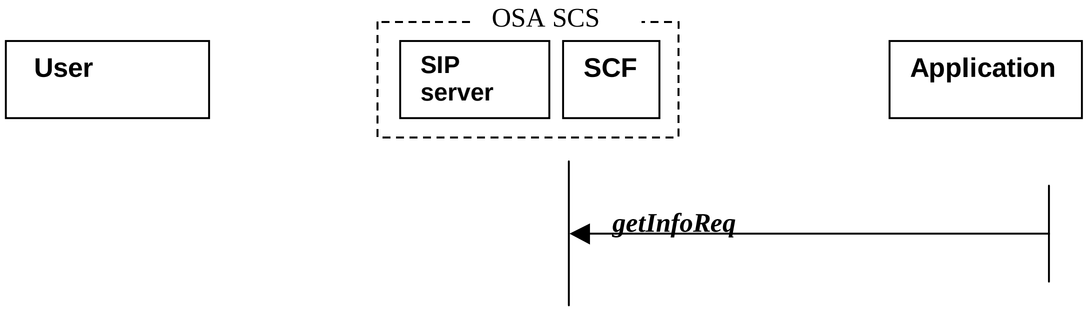

Figure 5-20: Call flow for getInfoReq()

Table 5-37: Normal operation

SIP Server Mode for the OSA SCS:	Proxy, UA, B2BUA or 3rd party controller (Any, except Redirect mode)
Pre-conditions:	**A relationship between the application and the call including associated objects exists. The *getInfoReq* method must be invoked before the call is routed to a target address.**
1	The application invokes the ***getInfoReq*** method. The SCS monitors the call to be capable to collect the requested information.
2	The OSA SCS will later on send the corresponding ***getInfoRes()*** or ***getInfoErr()*** based on the messages received from the SIP server of the OSA SCS.
3	
NOTE: The getInfoReq() method is not related to SIP signalling, it is sent by the application to request information associated to the call. Restriction: The getInfoReq method is only applicable on call level for a plain user initiated call between a caller and a callee, where a report is demanded when the destination leg or party (callee) terminates or when the call ends. (For application initiated calls and multiparty calls the method should instead be applied on a per destination leg (per callee)).	

Table 5-38: Parameter mapping

From: *getInfoReq*	**To: SIP**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	No direct mapping, merely a correlation is created.
callInfoRequested (TpCallInfoType) :	See table 6-10: **TpCallInfoType** mapping to SIP	
NOTE: There is no direct mapping to SIP. The getInfoReq() method results in supervision of the following SIP events via the SIP server of the OSA SCS: a) receipt of a SIP response ("answer" 200 OK/ACK) to an incoming INVITE request or b) the termination of an establishment dialog session (BYE)		

5.3.7 superviseReq

superviseReq (callSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : void

This method is called by the application to supervise a call.
The application can set a granted connection time for this call. If an application calls this method before it routes a call the time measurement will start as soon as the call is confirmed (answered) by the called party.

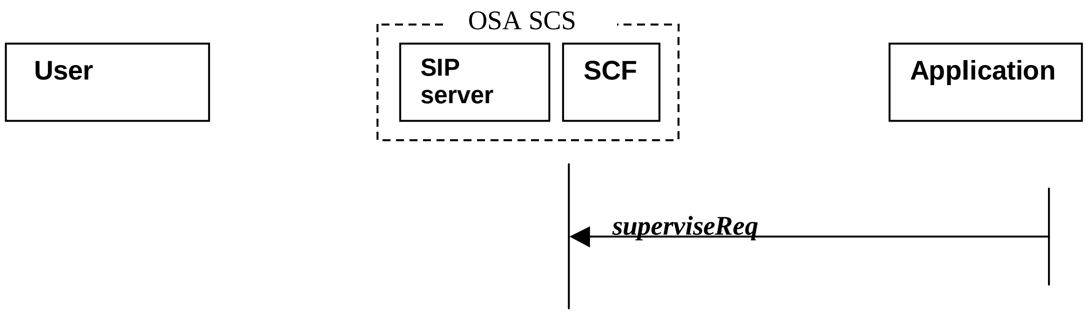

Figure 5-21: Call flow for superviseReq()

Table 5-39: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller** **(Any, except Redirect mode).** **However, if treatment (TpCallSuperviseTreatment) implies call release, then a UA mode of operation is demanded (UA, B2BUA, 3rd party controller). For this treatment, if the SCS is acting as a proxy, the only SIP message the SCS can generate after receiving superviseRes() in the call leg is BYE.**
Pre-conditions:	**A relationship between the application and the call including associated objects exists. The *superviseReq* method must be invoked before the call is confirmed, i.e. before answered.**
1	The application invokes the ***superviseReq*** method. The SCS monitors the call to be capable to collect the requested information.
2	The OSA SCS will later on send the corresponding ***superviseRes()*** or ***superviseErr()*** based on the messages received from the SIP server of the OSA SCS.
NOTE: The SIP server of the OSA SCS should use the messages received by the SIP server during the call session in order to sent the corresponding *superviseRes()* or s*uperviseErr()* method.	

Table 5-40: Parameter mapping

From: *getInfoReq*	**To: SIP**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation .
time (TpDuration)	**ACK** (confirmation of "answer" SIP 200 OK)	No direct mapping , but specified call supervision timer is to start upon the confirmation of answer event.
treatment (TpCallSuperviseTreatment) :	N/A See Note:	No direct mapping. Defines the treatment of the call by the call control service when the call supervision timer expires, e.g. release call (BYE) and /or send warning tone to calling party.
NOTE: There is no direct mapping to SIP. However, the expiry of the call supervistion timer during the active call initiates the action as specified in TpCallSuperviseTreatment.		

5.3.8 setAdviceOfCharge

setAdviceOfCharge (callSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : void

This method allows the application to determine the charging information that will be send to the end-users terminal.

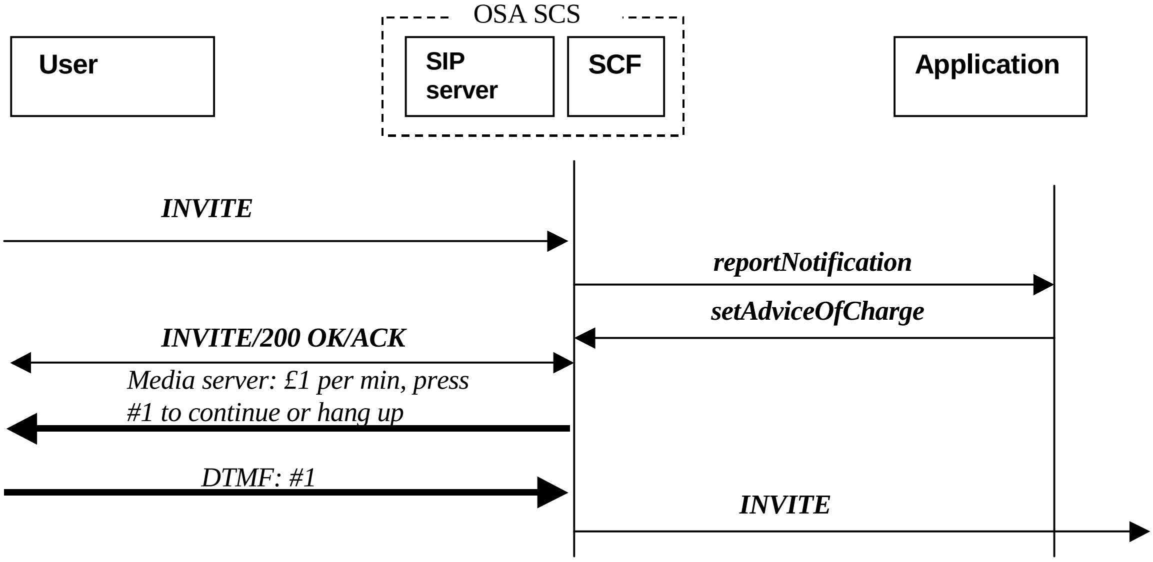

Figure 5-22: Call flow for setAdviceOfCharge()

Table 5-41: Normal operation

SIP Server Mode for the OSA SCS:	**UA mode** **The generation of a SIP message on request from the application demands the SIP server of the OSA to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).** **The SCS's behaviour on receiving setAdviceOfCharge is not standardized, the diagram above is just shown as an example on how this can be done.**
Pre-conditions:	**A relationship between the application and the call including associated objects exists. The *setAdviseOfCharge* method must be invoked before the call is confirmed, i.e. before answered.**
1	The application invokes the ***setAdviceOfCharge*** method. The SCS enables the call to be capable to send the requested information to the end-user.
2	
NOTE: How the SIP server of the OSA SCS sent the information to the calling party is not standardized in this release.	

Table 5-42: Parameter mapping

From: *setAdviceOfCharge*	**To: SIP**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation .
aOCInfo (TpAoCInfo): - ChargeOrder (**TpAoCOrder**) - Currency (**TpString**)	See Table 6-19 TpAoCInfo mapping to SIP.	Currency unit according to ISO-4217:1995 [8]
tariffSwitch (TpDuration)	N/A	

5.3.9 SetChargePlan

setChargePlan (callSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : void

This is a method that allows the application to set an operator specific charge plan for the call enabling to include charging information in network generated CDR.

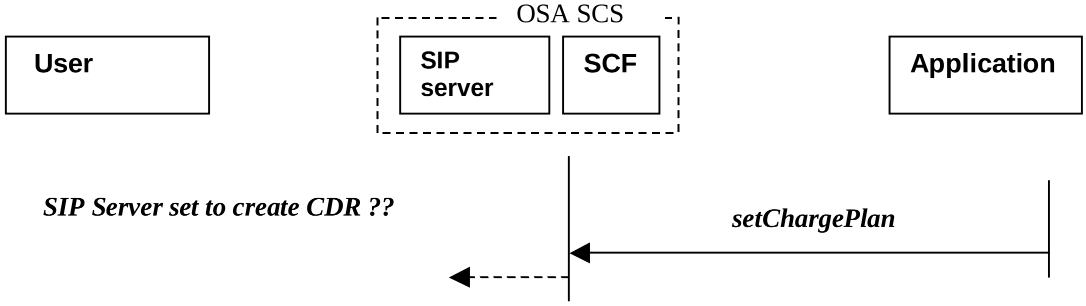

Figure 5-23: Call flow for setChargePlan()

Table 5-43: Normal operation

SIP Server Mode for the OSA SCS:	**Any mode.** **For details on application server handling IMS charging, see 3GPP TS 23.218 [6].**
Pre-conditions:	**A relationship between the application and the call including associated objects exists. The *setChargePlan* method may have to be invoked before the call is confirmed, i.e. before answered .**
1	The application invokes the ***setChargePlan*** method. The SCS enables the call to be capable to be charged according to defined plan .
2	
NOTE: The SIP server of the OSA SCS should invoke the requested charge plan. Information relevant to application and SCS not to SIP signalling.	

Table 5-44: Parameter mapping

From: *setChargePlan*	**To: SIP**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
callChargePlan (TpCallChargePlan)	N/A	Information relevant to application and SCS not to signalling

5.4 Multi-Party Call Application Interface

5.4.1 createAndRouteCallLegErr

createAndRouteCallLegErr (callSessionID : in TpSessionID, callLegReference : in TpCallLegIdentifier, errorIndication : in TpCallError) : void

This method is an asynchronous method which indicates that the request to route the call to the destination party was unsuccessful – the call could not be routed to the destination party (for example, parameters were incorrect, invalid address, the request was refused, etc).

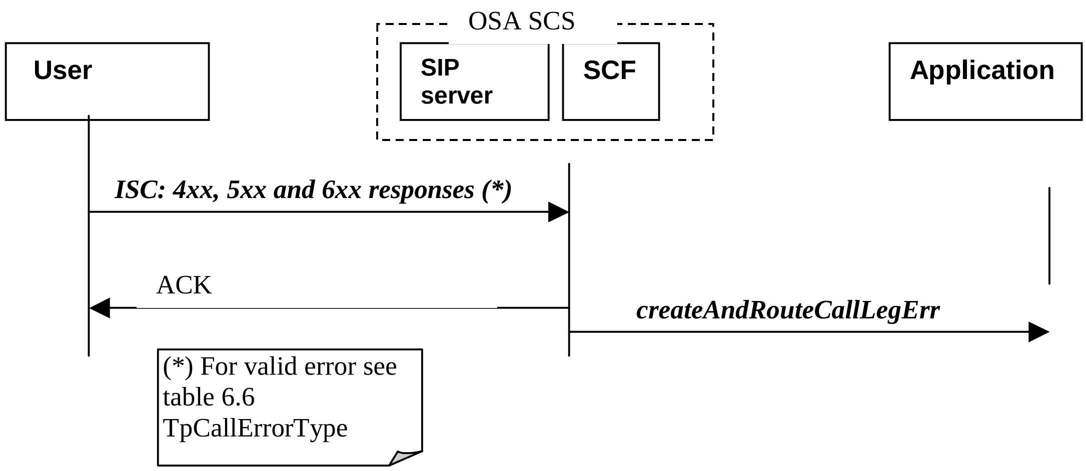

Figure 5-24: Call flow for *createAndRouteCallLegErr()*

Table 5-45: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller (Any, except Redirect mode.)**
Pre-conditions:	**Application has sent createAndRouteCallLegReq() , a request to route the call to the destination party.**
1	The request is refused e.g. the SIP server in the core network detects an error and notifies the SIP server of the SCS.
2	The SCS invokes the ***createAndRouteCallLegErr*** method
NOTE: The SIP server of the OSA SCS should detect the denial.	

Table 5-46: Parameter mapping

To: *createAndRouteCallLegErr*	**From: SIP**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation .
callLegReference (TpCallLegIdentifier)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	
errorIndication (TpCallError)	See table 6-5: **TpCallError** mapping from SIP	

5.4.2 callEnded

callEnded (callSessionID : in TpSessionID, report : in TpCallEndedReport) : void

This method is invoked when the call has terminated in the network. Furthermore, the operation contains an indication on the reason why the call has been ended. The method will always be invoked when the call is ended.

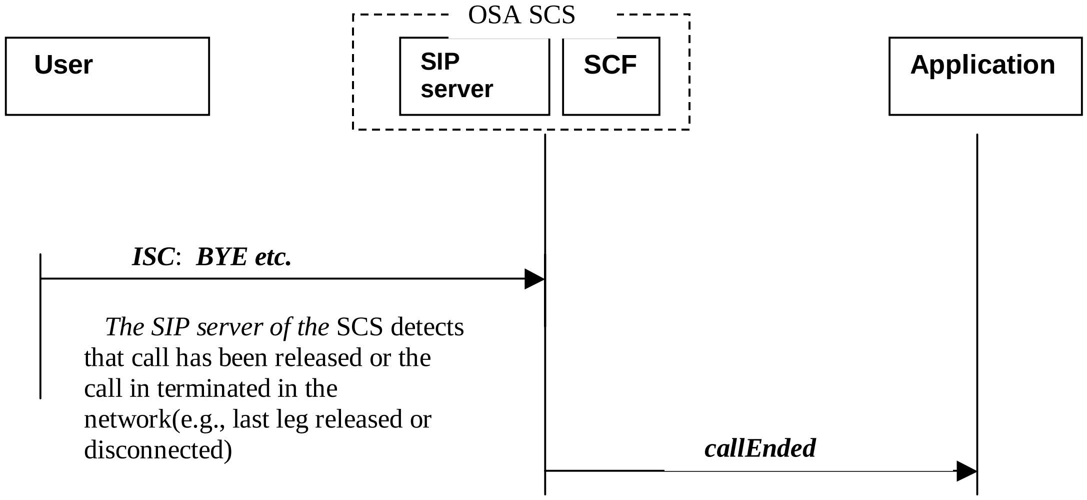

Figure 5-25: Call flow for *callEnded()*

Table 5-47: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect. (Any)**
Pre-conditions:	**There is an application monitoring the call in some way.**
1	The SCS detects that there is no leg connected to the call or the call has been released. The SCS invokes the ***callEnded*** method.
NOTE: The *callEnded()* method is sent to the application when the last leg has released or the call itself was released or no party has answered the call. This method does not require any SIP mapping. It reflects the call state in the SCS.	

Table 5-48: Parameter mapping

To: *callEnded*	**From: SIP: BYE, 3xx, 4xx, 5xx, 6xx**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	No direct mapping – a correlation.
report (TpCallEndedReport) :	-	
- CallLegSessionID *(TpSessionID)*	See "OSA Call and SIP Dialogue Correlation Tables" Table 4-1 to 4-5.	
- Cause (TpReleaseCause)	See table 6-18: **TpReleaseCause** for the mapping from SIP	

5.4.3 getInfoRes

getInfoRes (callSessionID : in TpSessionID, callInfoReport : in TpCallInfoReport) : void

This is an asynchronous method that reports all the necessary information requested by the application, for example to calculate charging.

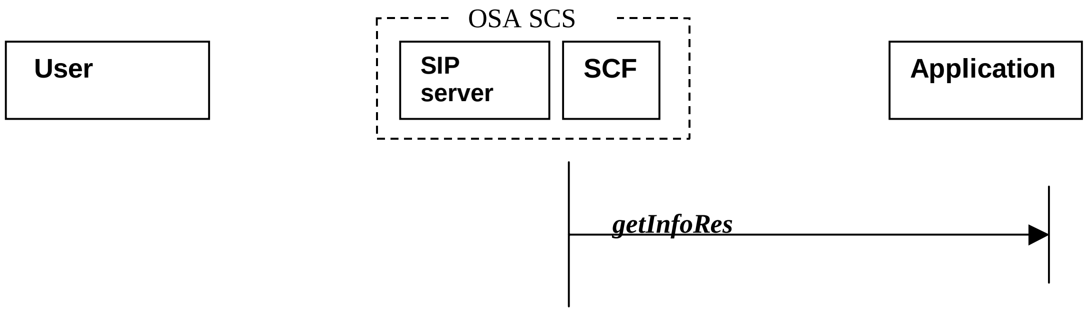

Figure 5-26: Call flow for *getInfoRes()*

Table 5-49: Normal operation

SIP Server Mode for the OSA SCS:	**(Proxy, UA, B2BUA or 3rd party controller)** **(Any, except Redirect mode)**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the** *getInfo**Req*** **method**
1	The OSA SCS detects that the call is terminated. The SCS invokes the ***getInfoRes()*** method

Table 5-50: Parameter mapping

To: *getInfoRes*	**From: SIP: BYE, 3xx, 4xx, 5xx, 6xx**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
callInfoReport (TpCallInfoReport):	-	
- CallInfoType (TpCallInfoType)	See Table 6-10: **TpCallInfoType**	Defines the type of call information requested and reported
- CallInitiationStartTime (TpDateAndTime)	N/A	The time when the SIP server of the OSA SCS sent the SIP INVITE message.
- CallConnectedToResourceTime (TpDateAndTime)	N/A	
- CallConnectedToDestinationTime (TpTpDateAndTime)	N/A	The moment the party received the ACK message for the INVITE. This information may be provided by the OSA SCS.
- CallEndTime (TpDateAndTime)	N/A	Moment when SIP BYE message is sent to participant or received from the participant.. This information may be provided by the OSA SCS.
- Cause (TpReleaseCause)	See Table 6-18 **TpReleasecause** for the mapping from SIP	

5.4.4 getInfoErr

getInfoErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This method is an asynchronous method that reports that the original request was erroneous, or resulted in an error condition.

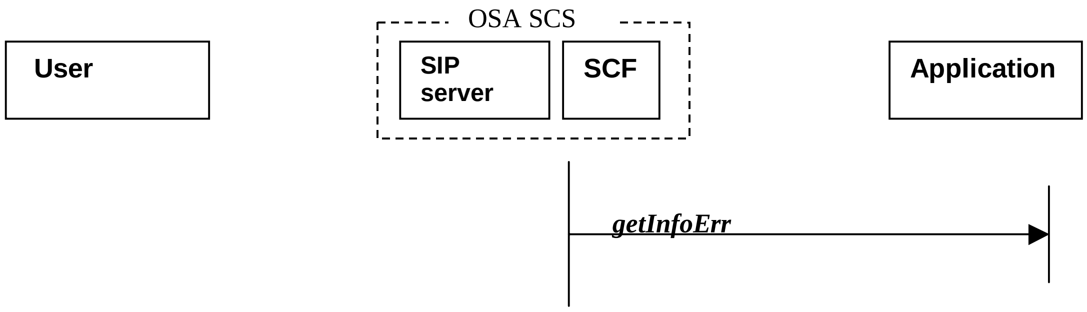

Figure 5-27: Call flow for *getInfoErr()*

Table 5-51: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any, except Redirect)**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the** *getInfoReq* **method**
1	The original request getInfoReq is erroneous or cannot be accepted due to e.g. call terminates abnormally.
2	The SCS identifies the correct applications that requested the call information and invokes the ***getInfoErr*** method.

Table 5-52: Parameter mapping

To: *getInfoErr*	**From: SIP 4xx**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError mapping** table from SIP.	

5.4.5 superviseErr

superviseErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This is an asynchronous method that reports a call supervision error to the application.

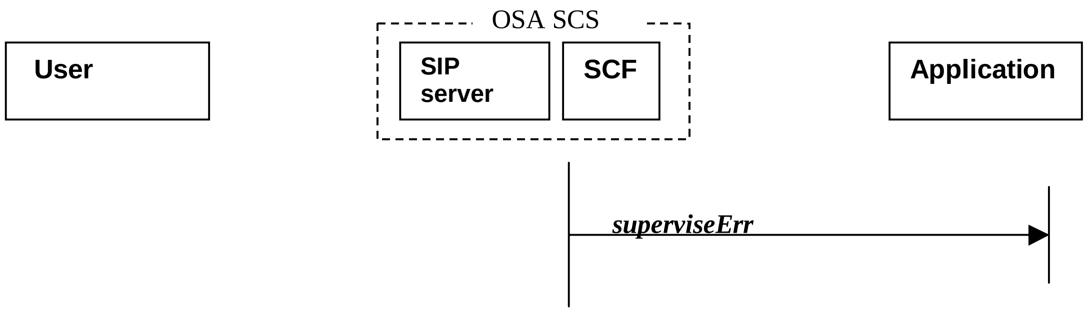

Figure 5-28: Call flow for *superviseErr()*

Table 5-53: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller (Any, except Redirect mode).** **However, if treatment (TpCallSuperviseTreatment) implies call release, then UA mode of operation is demanded. For this treatment, if the SCS is acting as a proxy, the only SIP message the SCS can generate after sending superviseErr() in the call leg is BYE.**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the *superviseReq* method.**
1	The SCS detects an error that can affect call supervision, e.g. call routing error.
2	The SCS identifies the correct applications that requested the call information and invokes the ***superviseErr*** method.

Table 5-54: Parameter mapping

To: *createAndRouteCallLegErr*	**From: SIP 4xx**	**Remark**
callSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation .
errorIndication (TpCallError)	See Table 6-5: **TpCallError** mapping from SIP	

5.4.6 superviseRes

superviseRes (callSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : void

This is an asynchronous method that reports a call supervision event to the application.

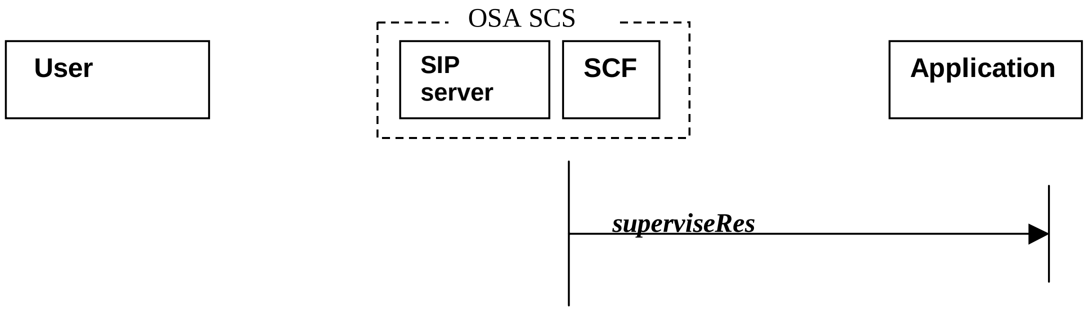

Figure 5-29: Call flow for *superviseRes()*

Table 5-55: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller (Any, except Redirect mode).** **However, if treatment (TpCallSuperviseTreatment) implies call release, then UA mode of operation is demanded. For this treatment, if the SCS is acting as a proxy, the only SIP message the SCS can generate after sending superviseErr() in the call leg is BYE.**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the *superviseReq* method. The specified call supervision timer expires.**
1	The OSA SCS detects that the supervision time is expired and acts according to the requested treatment (e.g. release call sending BYE) in ***superviseReq*** The OSA SCS identifies the correct application and invokes the ***superviseRes*** method.

Table 5-56: Parameter mapping

To: *superviseRes*	From: SIP	Remark
callSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation .
report (TpCallSuperviseReport)	N/A	Defines the response(s) from the call control service for calls that have been supervised, (e.g. timeout, call-ended, tone-applied, UI-finished).
usedTime (TpDuration)	N/A	No direct mapping to SIP:

5.5 CallLeg Service Interface

The call leg interface class represents the logical call leg associating a call with an address.
The leg represents the signalling relationship between the call and an address.

5.5.1 routeReq

routeReq (callLegSessionID : in TpSessionID, targetAddess : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, connectionProperties : in TpCallLegConnectionProperties) : void

This method is an asynchronous method used to request routing of the call leg to the remote party indicated by the target address.

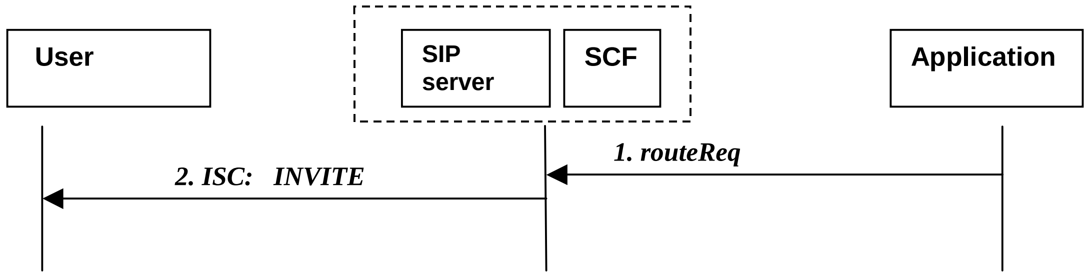

Figure 5-30: Call flow for routeReq(), UA mode

5.5.1.1 Case 1 UA mode operation

Table 5-57: Normal operation, UA operation mode

SIP Server Mode for the OSA SCS:	**UA mode** **The generation of a SIP message (INVITE) on request from the application demands the SIP server of the OSA to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	**A relationship between the application and the call including associated objects exists.** **For the routeReq() method, the SCS does not create any new call or call leg objects since the method is called on the existing Terminating Call Leg object**
1	The application invokes the ***routeReq*** method. The SCS enables the call to be set-up by issuing an invitation (INVITE) for the end-user to be called.
2	
NOTE 1: The routeReq method is applicable only for the terminating leg in the MPCC call leg STD. The SIP server of the OSA SCS should sent the INVITE for request thee routing to remote party. Forking is not supported by the OSA API. The call flow for this method is the equivalent to the *createCallAndRouteReq()* method. NOTE 2: When operation in B2BUA mode the flow is similar to UA mode, but behaviour reflects a specialisation of a proxy server comprising the split of the SIP dialogue between the end-users into two dialogues – one for each call party enabling the application to gain full session control. NOTE 3: See also Annex B and the flow examples B2-B5.	

Table 5-58: Parameter mapping, UA mode operation

From: *routeReq*	**To: SIP INVITE**	**Remark**
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation is created.
targetAddress (TpAddress)	**Request-URI** See Table 6-2: **TpAddress** mapping to SIP.	
originatingAddress (TpAddress)	**FROM header:** **SIP URL** See Table 6-2: **TpAddress** mapping to SIP.	
appInfo (TpCallAppInfoSet)	See Table 6-4: **TpCallAppInfo** mapping to SIP.	
ConnectionProperties (TpCallLegConnectionProperties):	See Table 6-12 **TpCallLegConnectionProperties** mapping to SIP.	
NOTE: See also Annex B and Annex C.		

5.5.1.2 Case 2 Proxy mode operation

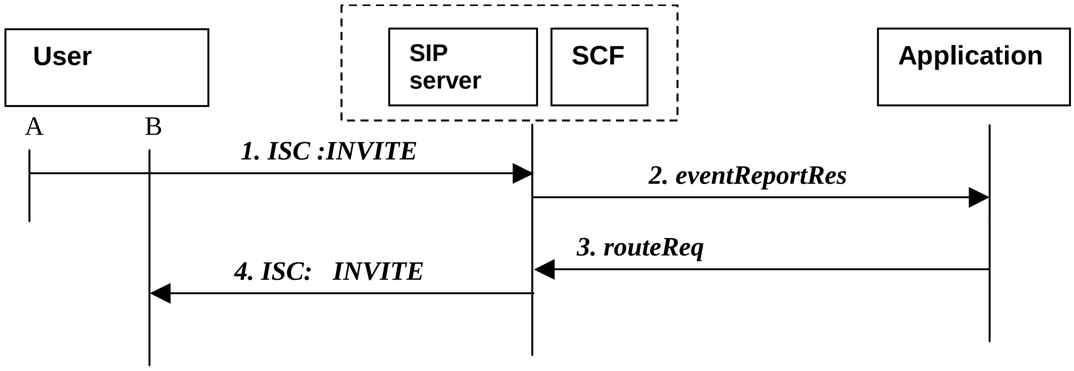

Figure 5-31: Call flow for routeReq(), Proxy mode

Table 5-59: Normal operation, Proxy operation mode

SIP Server Mode for the OSA SCS:	**Proxy mode** **The routeReq is used to forward a call (SIP message (INVITE)) on request from the application: The SIP server of the OSA SCS operates in proxy mode.**
Pre-conditions:	**A relationship between the application and the call including associated objects exists.** **For the routeReq() method, the SCS does not create any new call or call leg objects since the method is called on the terminating call leg object**
1	The application invokes the ***routeReq*** method. The SCS enables the call to be set-up by proxying the invitation (INVITE) for the end-user to be called.
NOTE: The routeReq method is applicable only for the terminating leg in the MPCC call leg STD. The SIP server of the OSA SCS should forward sent the INVITE for request the routing to remote party. Forking is not supported by the OSA API. The call flow for this method is equivalent to createCallAndRouteReq() method.	

Table 5-60: Parameter mapping, Proxy mode operation

From: *routeReq*	**To: SIP INVITE**	**Remark**
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-2.	No direct mapping – a correlation is created.
targetAddress (TpAddress)	**Request-URI Header or P-Called-Party-ID** [19] See Table 6-2: **TpAddress** mapping to SIP.	**When the SCS receives an INVITE (flow 1 in figure 5-31), if the P-Called-Party-ID header is present, then uses this header to identify the target address in the outgoing INVITE (flow 4 in figure 5-31). If not, then uses the Request-URI instead.**
originatingAddress (TpAddress)	N/A	**FROM header: not to be changed**
appInfo (TpCallAppInfoSet)	See Table 6-4: **TpCallAppInfo** mapping to SIP	
ConnectionProperties (TpCallLegConnectionProperties):	See Table 6-12: **TpCallLegConnectionProperties**	
NOTE: See also Annex B and Annex C.		

5.5.2 eventReportReq

eventReportReq (callLegSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet) : void

This method is an asynchronous method used to set, clear or change criteria for the events that the Call Leg object will observe.

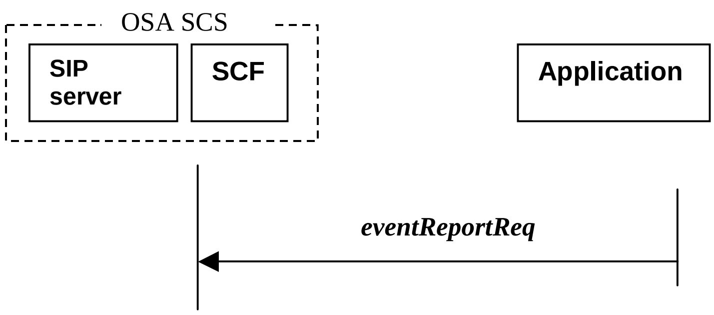

Figure 5-32: Call flow for eventReportReq()

Table 5-61: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any mode, except Redirection.)**
Pre-conditions:	**A relationship between the application and the call including associated leg objects exists. The *eventReportReq* method must be invoked before call set-up (e.g. *routeReq* method) if to monitor events reporting the results of the call set-up request (invitation).**
1	The application invokes the ***eventReportReq*** method. The OSA SCS enables the call to be monitored for subsequent events to be reported.
2	The SCS monitors the call and will later on send the corresponding ***eventReportRes()*** or ***eventReportErr()*** based on the messages received for the controlling entity, i.e. the SIP server of the OSA SCS.
NOTE: The ***eventReportReq*** method is applicable for any leg created leg being part of the MPCC call leg STD.	

Table 5-62: Parameter mapping

From: *eventReportReq*	**From: SIP**	**Remark**
callLegSessionID	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	A correlation - no direct mapping
eventsRequested (TpCallEventRequestSet)	See Table 6-8: **TpCallEventRequest** mapping from SIP.	

5.5.3 release

release (callLegSessionID : in TpSessionID, cause : in TpReleaseCause) : void

This method is used to request the release of a single call leg.

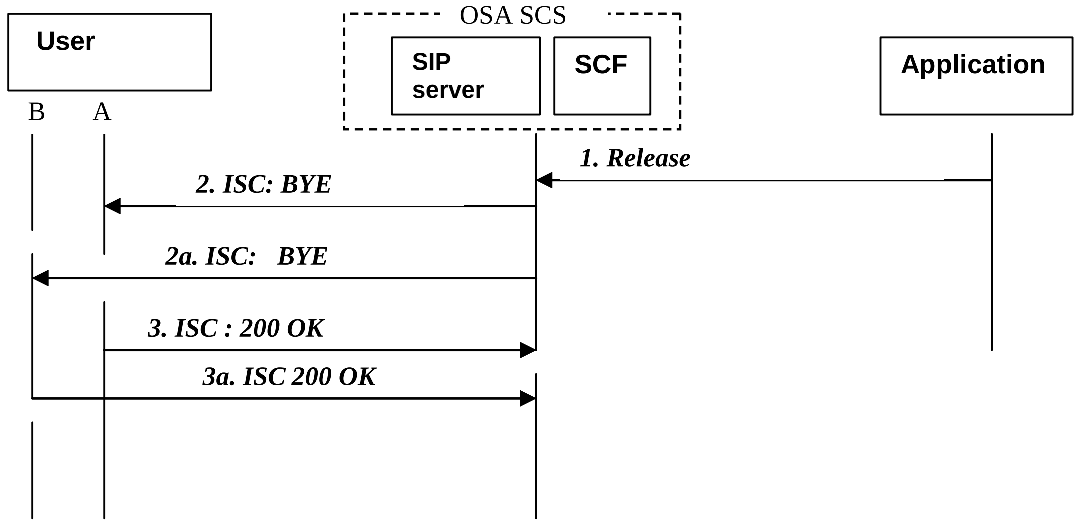

Figure 5-18a: Call flow for *release*, acting as proxy

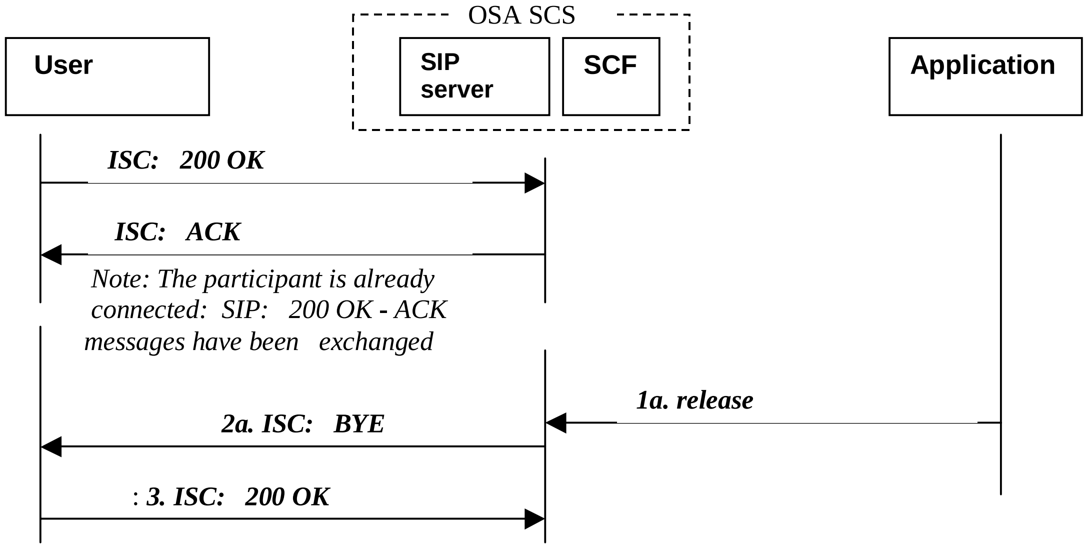

Figure 5-33: Scenario a: Call flow for *release()*, participant connected, UA mode

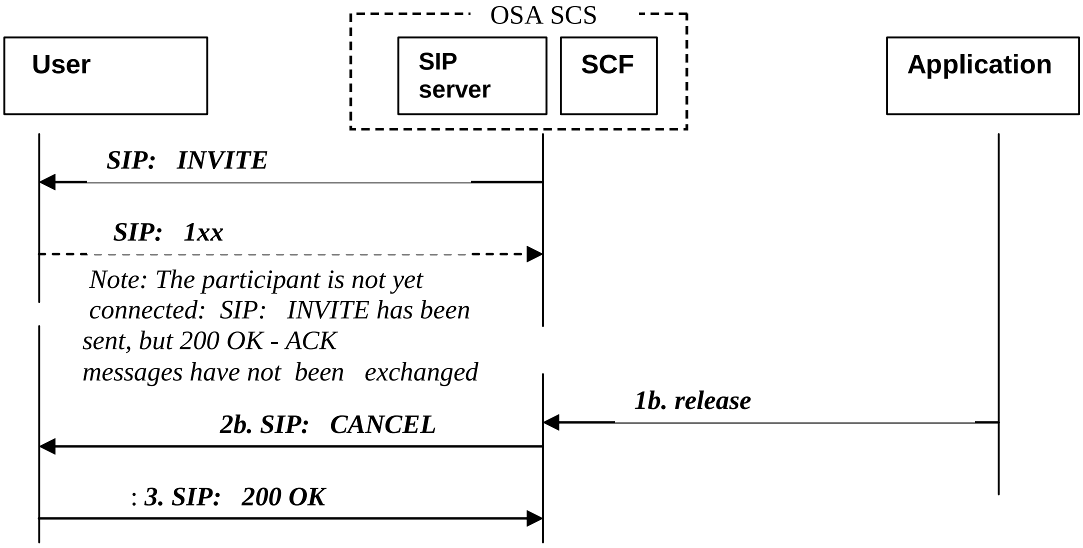

Figure 5-34: Scenario b: Call Flow for *release(),* pending call attempt toward participant, UA mode

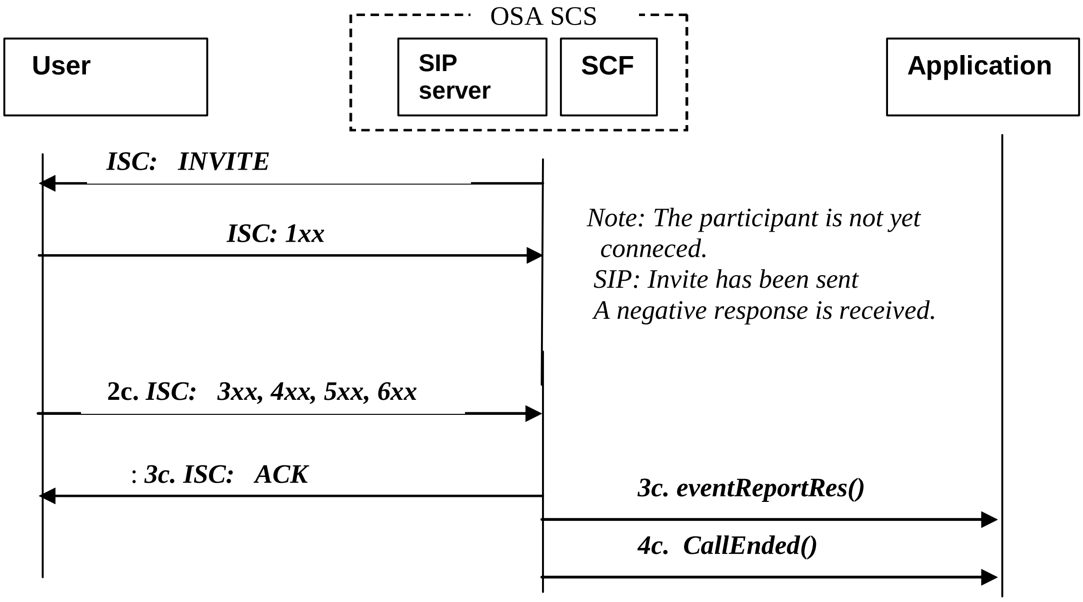

Figure 5-35: Scenario c: Call flow for *release(),* call (invite) to participant not accepted

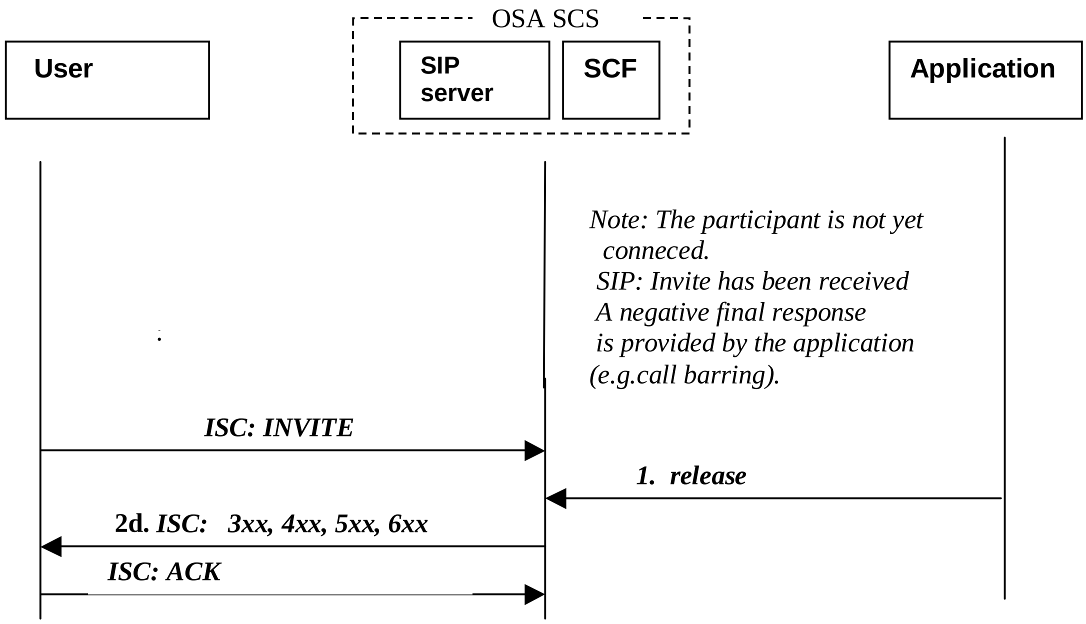

Figure 5-36: Scenario d: Call flow for *release()*, call (invite) from participant not accepted

Table 5-63: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy and UA mode** **The generation of a SIP message (BYE) on request from the application to release participants in the call demands the SIP server of the OSA to operate in a proxy or UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	Call is in progress
1	The application or the SCS invokes the ***release*** method. The SCS generates the SIP message to release the requested parties (call leg) from the call
2a	Scenario 2a: SIP BYE is sent. The SIP server sends the BYE Message toward the participant connected to the call.
2b	Scenario 2b: SIP CANCEL is sent to terminate a pending leg. The SIP server sends the CANCEL message toward the participants associated to the call but not connected yet. Note: CANCEL secures in case of SIP forking that all with the OSA leg possible associated pending SIP legs will be released. CANCEL cannot be sent when SCS is acting as a proxy.
2c	Scenario 2c: The invitation to a participant is not accepted. The application sends a Release to terminate its leg. Note: It could also send a ***continueProcessing()*** or ***deassign()*** to terminate it logical call leg object representing the connection (SIP leg) in the network. !!
2d	
NOTE: For scenario 2c the application could instead of release() send a ***continueProcessing()*** or ***deassign()*** to terminate it logical call leg object representing the connection (SIP leg) in the network. !! When operating in B2BUA mode the decision whether a release from one participant will cause the release of any other participant can be controlled by the application.	

Table 5-64: Parameter mapping

From: *release*	**To: SIP** BYE, 4xx, 5xx, Cancel (if any pending INVITE requests from application)	**Remark**
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	A correlation - no direct mapping
cause (TpReleaseCause)	See table 6-17: **TpReleaseCause** for mapping to SIP	See table for **TpReleaseCause** for mapping to SIP response codes
NOTE: The release() method may be sent any time from the application e.g. resulting in a) the termination of an establishment session (BYE) or b) the cancellation of a pending request (CANCEL) after the application has issued an INVITE request. c) the termination of an unsuccesful call attempt (e.g. meeting busy, not reachable etc.) or d) creation of a SIP response (e.g. 4xx, 5xx) to an incoming INVITE request.		

5.5.4 getInfoReq

getInfoReq (callLegSessionID : in TpSessionID, callLegInfoRequested : in TpCallLegInfoType) : void

This method is an asynchronous method that requests information associated with the call to be provided at the appropriate time (for example, to calculate charging).

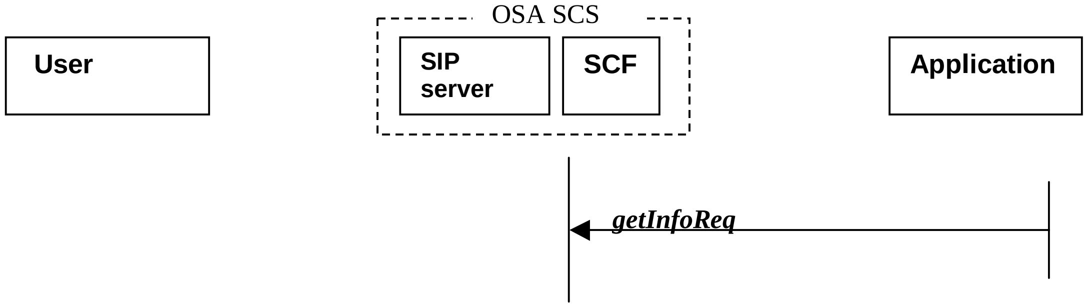

Figure 5-37: Call flow for *getInfoReq()*

Table 5-65: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3^{rd} party controller. (Any, except Redirect mode.)**
Pre-conditions:	**A relationship between the application and the call including associated call leg objects exists.** **The *getInfoReq* method must be invoked on a call leg before the call leg is routed to a target address.**
1	The application invokes the ***getInfoReq*** method. The SCS monitors the call leg to be capable to collect the requested information.
2	The OSA SCS will later on send the corresponding ***getInfoRes()*** or ***getInfoErr()*** based on the messages received from the SIP server of the OSA SCS.
3	
NOTE 1: The getInfoReq() method is not related to SIP signalling, it is sent by the application to request information associated to the call. Indeed the method does not involve SIP mapping. NOTE 2: The OSA SCS should use the messages received by the SIP server during the call session in order to sent the corresponding *getInfoRes()* or *getInfoErr()* method.	

Table 5-66: Parameter mapping

From: *getInfoReq*	To: SIP	Remark
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
callLegInfoRequested (TpCallLegInfoType)	See table 6-11: **TpCallLegInfoType**	
NOTE: There is no direct mapping to SIP. The getInfoReq() method results in supervision of the following SIP events: a) receipt of a SIP response (200 OK/ACK) to an incoming INVITE request or b) the termination of an establishment session (BYE).		

5.5.5 getCall

getCall (callLegSessionID : in TpSessionID) : TpMultiPartyCallIdentifier

This method used to retrieve the reference of the Call object associated with the Call leg object.

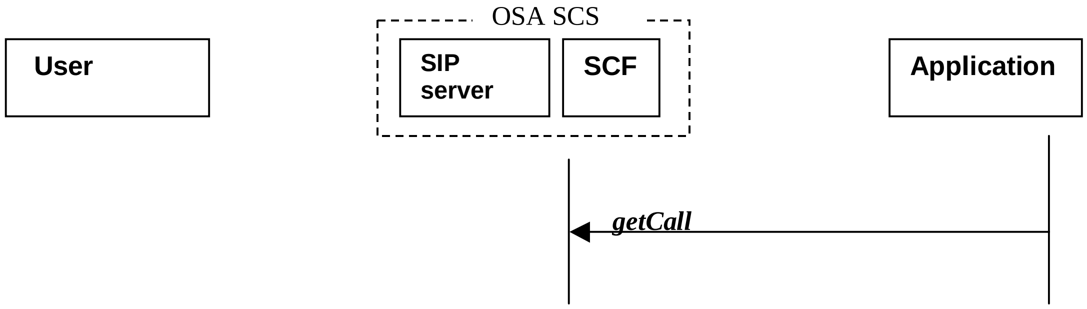

Figure 5-38: Call flow for *getCall()*

Table 5-67: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect. (Any)**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. The *getCall* method can be invoked on any existing call leg object.**
1	The application invokes the ***getCall*** method. The SCS return the associated call object reference to the application.
NOTE: The ***getCallLeg()*** method is not related to SIP signalling, it is sent by the application to request information about the associated logical call object in the SCS. Indeed the method does not involve any SIP mapping.	

Table 5-68: Parameter mapping

From: *getInfoReq*	To: SIP	Remark
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping, merely a correlation is created.
Returns: TpMultiPartyCallIdentifier **- CallReference** (IpMultiPartyCallRef) **- CallSessionID** (TpSessionID)	N/A	

5.5.6 continueProcessing

continueProcessing (callLegSessionID : in TpSessionID) : void

This method used to continue processing of the call.

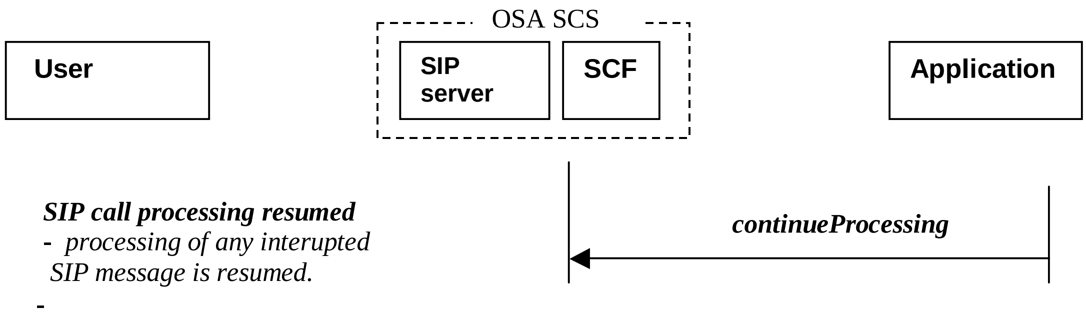

Figure 5-39: Call flow for *continueProcessing()*

Table 5-69: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller (Any, except Redirection.)**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. Call processing is suspended and the application is informed of call related events in interrupt mode.**
1	The application invokes the ***continueProcessing*** method requesting processing for the call leg object to be resumed.
2	The SCS requests the SIP server of the OSA SCS to resume SIP processing, when the call is to be resumed. That is the necessary response(s) from the application to resume call processing has been determined.
NOTE: The c***ontinueProcessing*** method is addressed to a single leg object. Resumption of SIP call processing occurs when all the MPCCS leg objects STDs are in processing state (not suspended). The *continueProcessing* method can be invoked on any existing call leg object to resume processing.	

Table 5-70: Parameter mapping

From: continueProcessing	To: SIP	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping, merely a correlation is created.

5.5.7 attachMediaReq

attachMediaReq (callLegSessionID : in TpSessionID) : void

This asynchronous method used to request that the call leg be attached to its call object. This will allow transmission on all associated bearer connections or media streams to and from other parties in the call. The call leg must be in the connected state for this method to complete successfully. However, the request may be sent as soon as the call leg object exists.

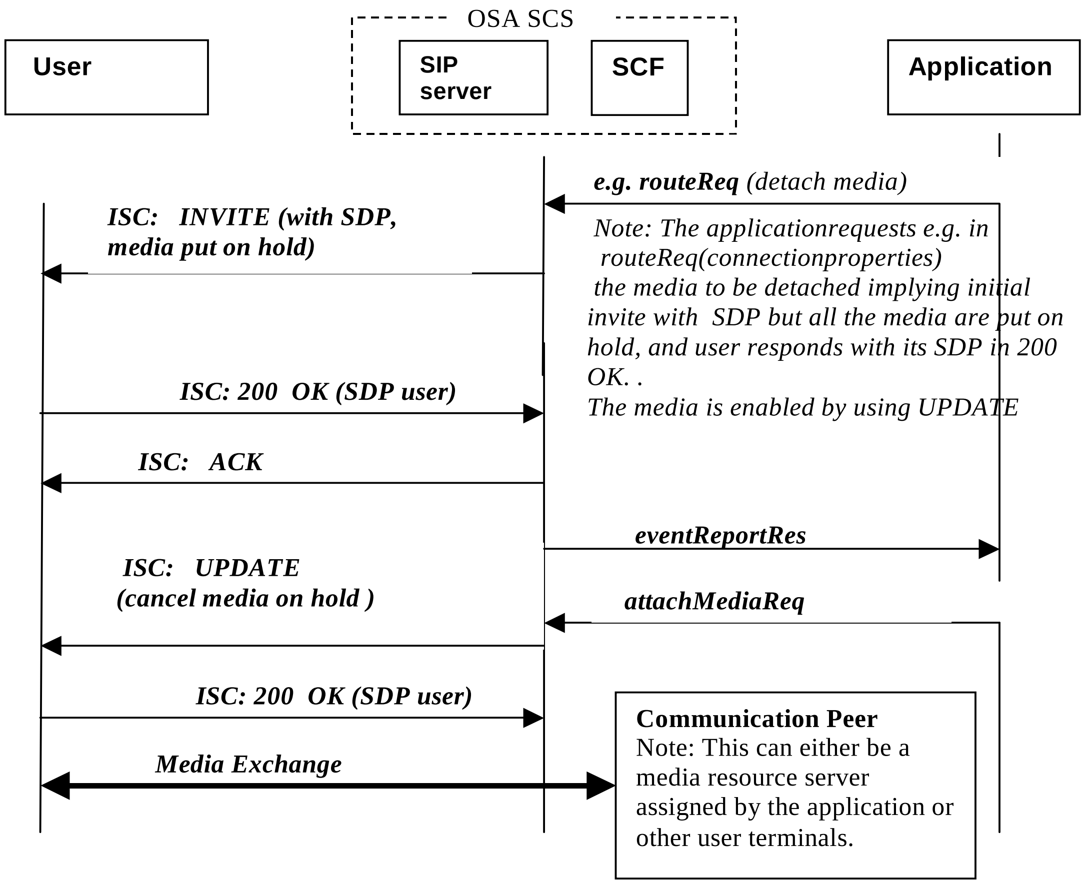

Figure 5-40: Scenario a: Call flow for *attachMediaReq(),* UA/B2BUA mode

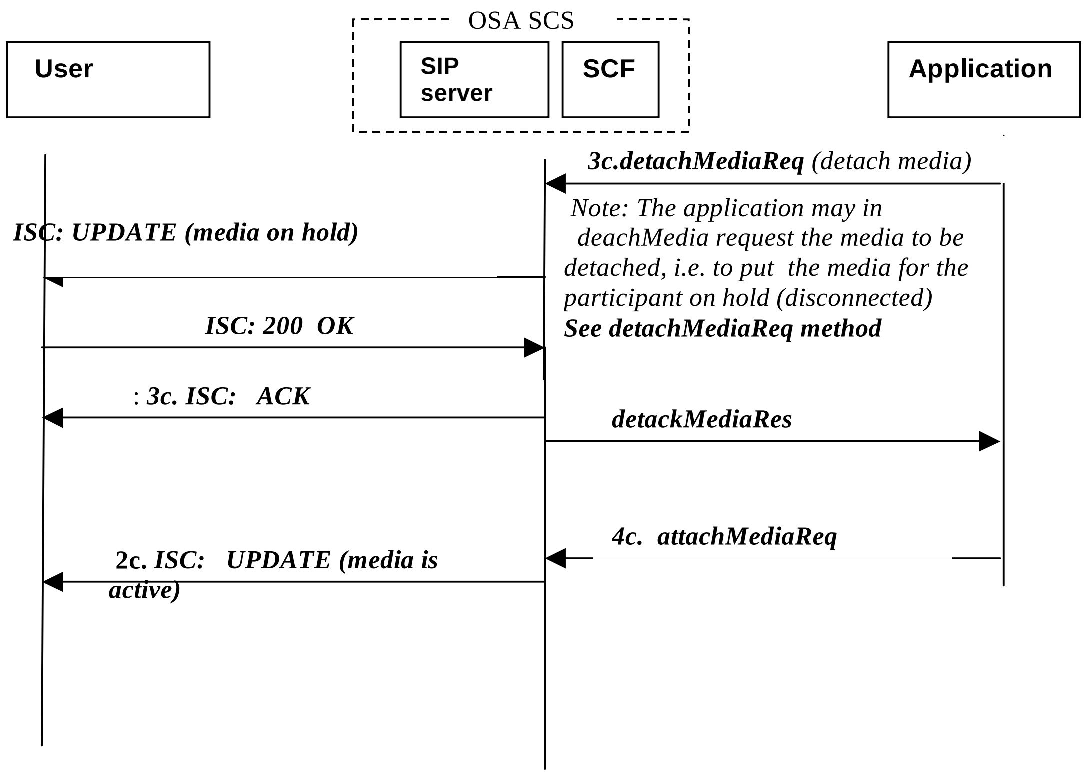

Figure 5-41: Scenario b: Call flow for *attachMediaReq(),* UA/B2BUA mode

Table 5-71: Normal operation

SIP Server Mode for the OSA SCS:	**UA, B2BUA, 3rd. party controller mode** **The generation of a SIP message (UPDATE [12]) on request from the application to attach media channels of a single user in the call demands the SIP server of the OSA SCS to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	**Call is processing. A relationship between the application and the call including associated call leg object(s) exists. The leg is in a connection state and has a media connection established with the others legs in the call.** **AttachMedia is not executed until the connected state is reached (200 OK /ACK) , i.e. if received before the SCS should buffer the request until it can be executed..**
1	The application invokes the ***attachMediaReq*** method requesting the media stream(s) for the call leg object to be attached, i.e. enabling media communication fie the call party. Application request the media attachment for this leg.
2	The SCS requests the SIP server of the OSA SCS to attach the media when the call enables this.. The SCS generates a new SIP UPDATE message to be sent to the participant, i.e. in this case the ***attachMediaReq()*** method is mapped onto the UPDATE message.
NOTE 1: The new UPDATE sent to the participant does not affect a SIP dialog, it is only updating the previous SIP session since the SIP call-ID will be the same, only the SIP CSEQ will be higher to indicate that the media description has changed. NOTE 2: The *attachMediaReq* method can be invoked on any existing call leg object to request the media attachment. If SIP processing is in the call set-up phase, the request is buffered until it can be executed, i.e. it is not executed until the phase in call procession where it is applicable to connect media. Note: no error is reported in case media is already attached. NOTE 3: In SIP, the natural behaviour is to establish the media session once the signalling is established. In OSA a party can be disconnected (detachMediaReq) and re-connected (attachMediaReq) to a call. A way to map this functionality in SIP is to use the SDP on hold feature enabling putting the media streams on hold (detach media) while the session is established or after the establishment. When the application will request to attach the media, a new UPDATE will be sent to the participant with the media session description. NOTE 4: See also Annex B and flow example B6.	

Table 5-72: Parameter mapping

From: *continueProcessing*	To: SIP	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.

5.5.8 detachMediaReq

detachMedia (callLegSessionID : in TpSessionID) : void

This asynchronous method is used to detach the call leg from its call, i.e., this will prevent transmission on any associated bearer connections or media streams to and from other parties in the call. The call leg must be in the connected state for this method to complete successfully.

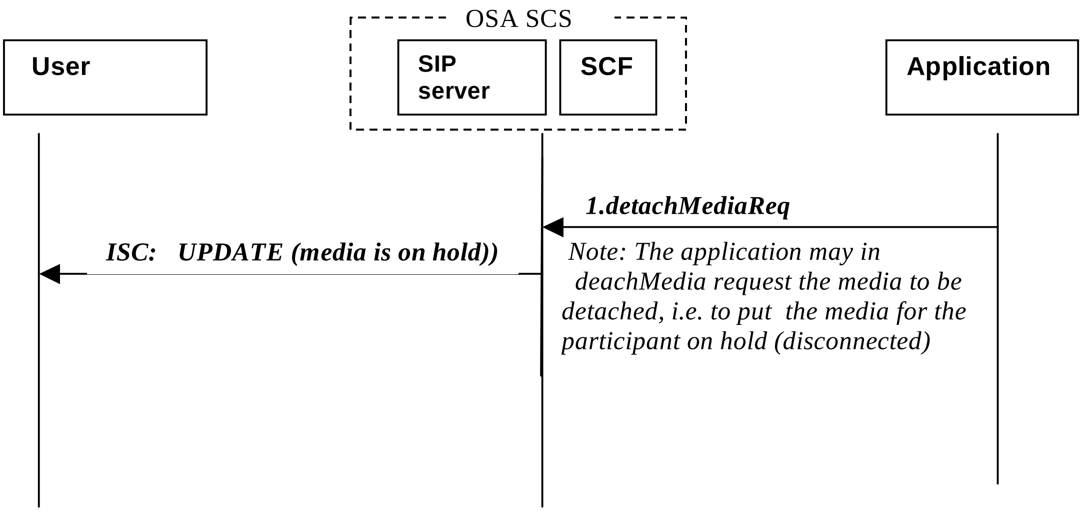

Figure 5-42: Call flow for *detachMediaReq(),* UA/B2BUA mode

Table 5-73: Normal operation

SIP Server Mode for the OSA SCS.	**UA, B2BUA, 3rd. party controller mode** **The generation of a SIP message (UPDATE) on request from the application to detach media channels of a single user in the call demands the SIP server of the OSA SCS to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	**Call is processing. A relationship between the application and the call including associated call leg object(s) exists. The leg is in a connection state and has a media connection established with the others legs in the call.** **DetachMedia is not executed until the connected state is reached (200 OK /ACK) , i.e. if received before the SCS should buffer the request until it can be executed.**
1	The application invokes the ***detachMediaReq*** method requesting the media stream(s) for the call leg object to be de-attached, i.e. enabling to put the media communication on hold for the call party. Application request the media de-attachment for this leg. The application prevents the transmission of media connection to this leg by calling the *detachMediaReq()*.
2	The SCS requests the SIP server of the OSA SCS to de-attach the media when the call enables this.. The SCS generates a new SIP UPDATE message to be sent to the participant, i.e. in this case the de*tachMediaReq()* method is mapped onto a SIP UPDATE message with an SDP on hold.
NOTE 1: The new UPDATE sent to the participant does not affect a SIP dialog, it is only updating the previous SIP session since the SIP call-ID will be the same, only the SIP CSEQ will be higher to indicate that the media description has changed. The ***detachMediaReq*** method can be invoked on any existing call leg object to request the media attachment. If SIP processing is in the call set-up phase, the request is buffered until it can be executed, i.e. it is not executed until the phase in call procession where it is applicable to connect media. Note: no error is reported in case media is already detached. In SIP, the natural behaviour is to establish the media session once the signalling is established. In OSA a party can be disconnected (detachMedia) and re-connected (attachMedia) to a call. A way to map this functionality in SIP is to use the SDP on hold feature enabling putting the media streams on hold (detach media) while the session is established or after the establishment. When the application will request to attach the media, a new UPDATE will be sent to the participant with the media session description. NOTE 2: See also Annex B and flow example B6.	

Table 5-74: Parameter mapping

From: *continueProcessing*	To: SIP	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.

5.5.9 deassign

deassignCall (callLegSessionID : in TpSessionID) : void

This method is used to request that the relationship between the application and the call leg and associated objects be de-assigned. It leaves the call in progress, however, it purges the specified call leg object so that the application has no further control of call leg processing. If a call leg is de-assigned that has event reports or call information reports requested, then these reports will be disabled and any related information discarded.

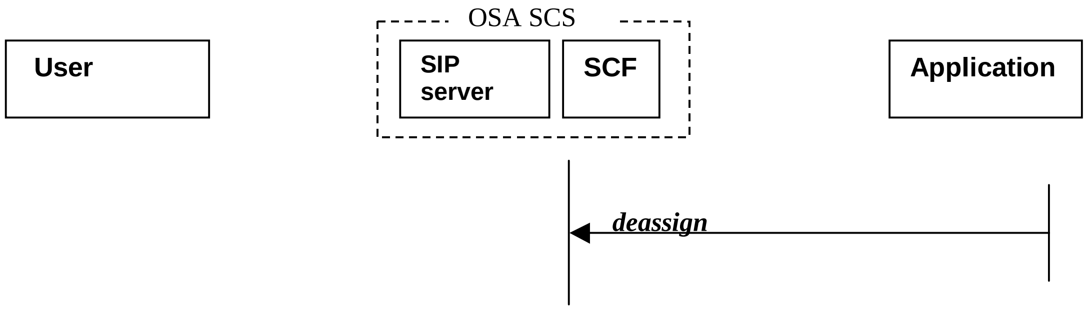

Figure 5-43: Call flow for *deassign()*

Table 5-75: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect. (Any)**
Pre-conditions:	**A relationship between the application and the call leg including associated objects exists.**
1	The application invokes the ***deassign*** method on a leg
2	The SCS terminates the relationship between the application and the call leg and its associated objects and notifies the SIP server of the OSA SCS.
3	The SIP server of the OSA SCS is to continue call processing autonomously, i.e. without any control from the application related to the call leg object. Any possible interrupted call processing related to the leg that has been deassigned control is to be resumed.
NOTE: If the application was the only one to control the session, the SIP server of the OSA SCS may remove itself from the route-request.	

Table 5-76: Parameter mapping

From: *continueProcessing*	To: SIP xx	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping, merely a correlation is created.

5.5.10 getCurrentDestinationAddress

getCurrentDestinationAddress (callLegSessionID : in TpSessionID) : TpAddress

This method is sent by the application to the leg to get the current address of the destination the leg has been directed to.

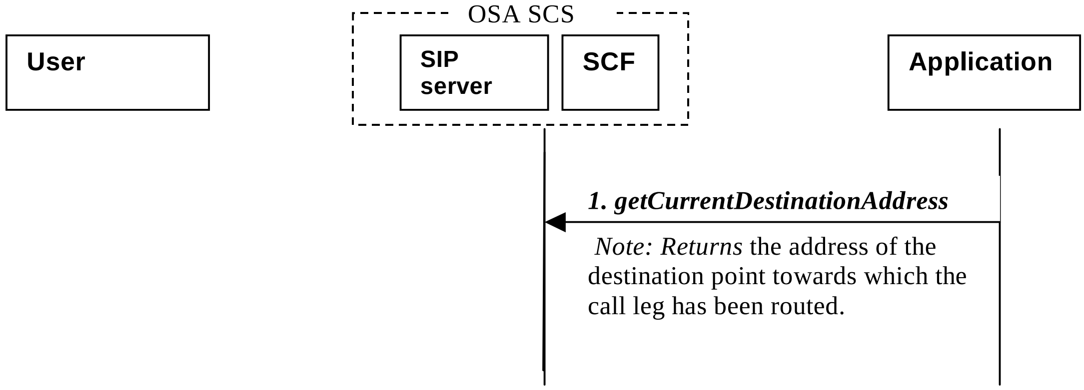

Figure 5-44: Call flow for getCurrentDestinationAddress()

Table 5-77: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any, except Redirect)**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. The leg is in a connection and is a terminating leg in the MPCCS STD.**
1	The application invokes the ***getCurrentDestinationAddress*** method requesting information for the call leg object regarding the address of current destination point..
2	The SCS returns the address of the destination point towards which the call leg has been routed in the method return parameter.
NOTE: The ***getCurrentDestinationAddress*** method can be invoked on any OSA MPCCS Terminating Call Leg object.	

Table 5-78: Parameter mapping

From: *getLastRedirectedAddress*	To: SIP	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping, merely a correlation is created..
Returns: **TpAddress**	See Table 6-2: **TpAddress** mapping to SIP.	Specifies the last address where the call leg was directed to.

5.6 CallLeg Application Interface

5.6.1 routeErr

routeErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This method is an asynchronous method which indicates that the request to route the call to the destination party was unsuccessful – the call could not be routed to the destination party (for example, parameters were incorrect, invalid address, the request was refused, etc).

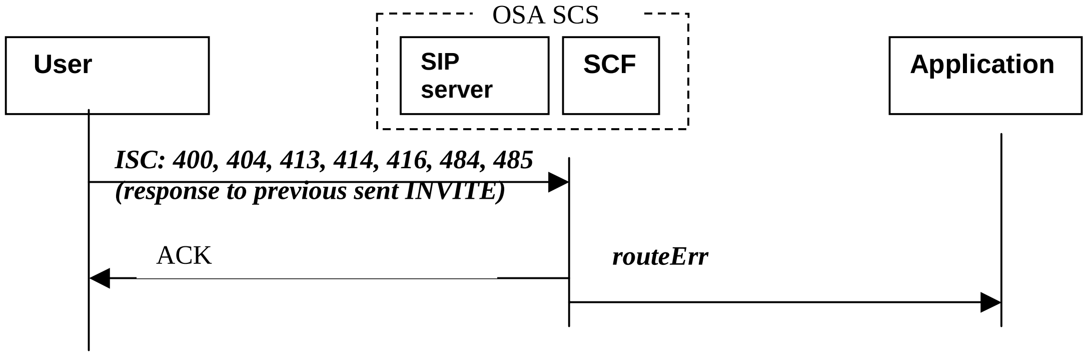

Figure 5-45: Call flow for *routeErr()*

Table 5-79: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any , except Redirect mode.)**
Pre-conditions:	**Application has sent** routeReq() **, a request to route the call to the destination party.**
1	The request is refused e.g. the SIP server in the core network detects an error and notifies the SIP server of the SCS.
2	The SCS invokes the ***routeErr*** method
NOTE: The SIP server of the OSA SCS could detect the denial.	

Table 5-80: Parameter mapping

To: *routeErr*	**From: SIP**	**Remark**
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError** for mapping from SIP.	

5.6.2 eventReportRes

eventReportRes (callLegSessionID : in TpSessionID, eventInfo : in TpCallEventInfo) : void

This asynchronous method is used to report that an event has occurred on the call leg that was requested to be reported (for example , a mid-call event from the party; the party has requested to disconnect; etc.).

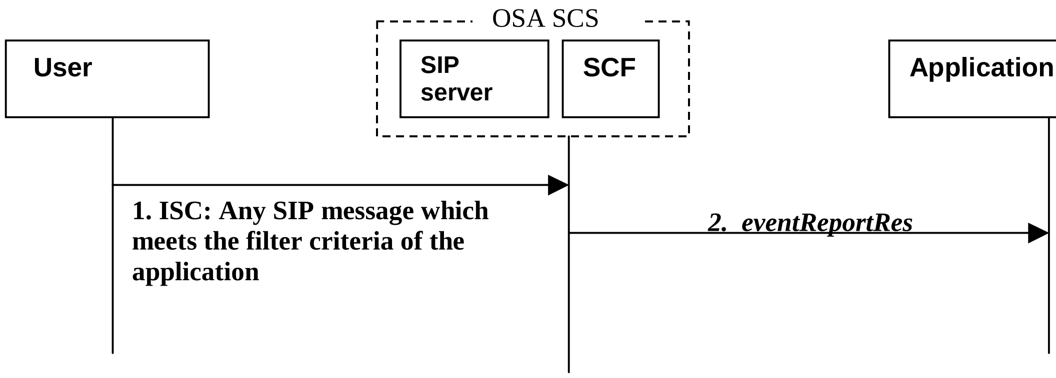

Figure 5-46: Call flow for *eventReportRes()*

Table 5-81: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller.**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. The application requested to be notified of the event with e.g. eventReportReq and this specific event has occurred in the network.**
1	The SIP server of the OSA SCS detects a SIP message (response or request) that corresponds to a requested call event to be reported to the application.
2	The OSA SCS invokes the ***eventReportRes()*** method.

Table 5-82: Parameter mapping

To: ***eventReportRes***	From: SIP (any SIP message)	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
eventInfo (TpCallEventInfo)	See Table 6-7: **TpCallEventInfo** mapping from SIP.	

5.6.3 eventReportErr

eventReportErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This method is an asynchronous method used to indicate that the request to manage call leg event reports was unsuccessful (for example, parameters were incorrect, the request was refused, etc).

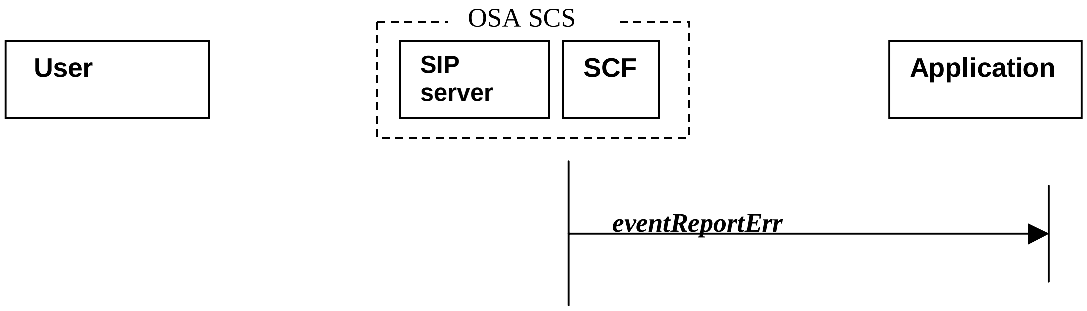

Figure 5-47: Call flow for *eventReportErr()*

Table 5-83: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller. (Any, except Redirect)**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the** *eventReportReq* **method**
1	The original request eventReportReq is erroneous - or cannot be accepted due to e.g. call terminates abnormally.
2	The SCS identifies the correct applications that requested the event report information and invokes the ***eventReportErr*** method.

Table 5-84: Parameter mapping

To: *eventReportErr*	**From: SIP 4xx**	**Remark**
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError** for mapping from SIP.	

5.6.4 callLegEnded

callLegEnded (callLegSessionID : in TpSessionID, cause : in TpReleaseCause) : void

This method is used to indicate to the application that the leg has terminated in the network. The application has received all requested results (e.g., getInfoRes) related to the call leg. The call leg will be destroyed after returning from this method. Furthermore, the operation contains an indication on the reason why the call leg has been ended. The method will always be invoked when the call leg is ended.

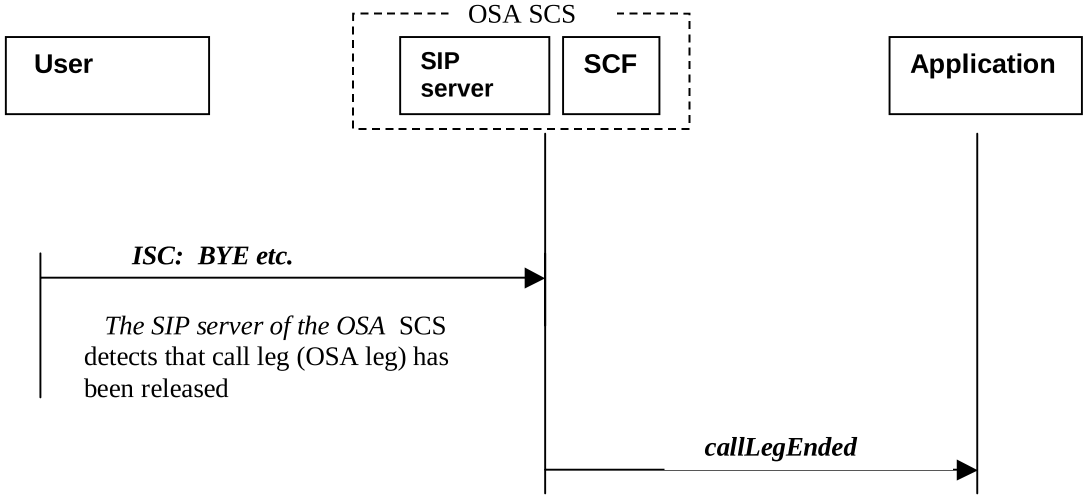

Figure 5-48: Call flow for *callLegEnded()*

Table 5-85: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect**
Pre-conditions:	**There is an application monitoring the call in some way.**
1	The SCS detects that the OSA call leg object connected to the call is destroyed, i.e. the call has been released. The SCS invokes the ***callLegEnded*** method.
NOTE: The *callLegEnd()* method is sent to the application when the party associated with the leg has released or the call itself was released to connection to the party .	

Table 5-86: Parameter mapping

To: *callLegEnded*	**From: SIP**	**Remark**
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping, merely a correlation is created
cause (TpReleaseCause)	See Table 6-18; **TpReleaseCause** Mapping from SIP	

5.6.5 getInfoRes

getInfoRes (callLegSessionID : in TpSessionID, callLegInfoReport : in TpCallLegInfoReport) : void

This is an asynchronous method that is used to report all the necessary information requested by the application, for example to calculate charging.

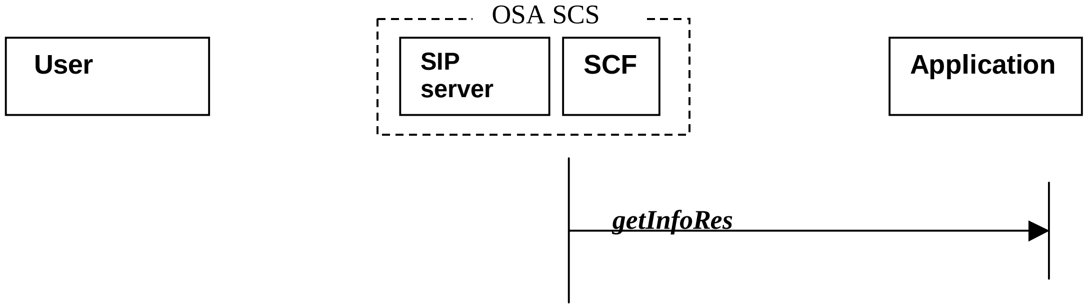

Figure 5-49: Call flow for *getInfoRes()*

Table 5-87: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect (Any)**
Pre-conditions:	Call is in progress. The application has requested call leg information with the ***getInfoReq*** method.
1	The SCS detects that the OSA call leg is terminated. The SCS invokes the ***getInfoRes()*** method. The OSA SCS has via its SIP Server collected the requested call related information which is reported to the application.

Table 5-88: Parameter mapping

To: *getInfoRes*	**From: SIP:**	**Remark**
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
callLegInfoReport (TpCallLegInfoReport):	**-**	
-CallLegInfoType (TpCallLegInfoType)	**N/A**	Indicates the type of the call leg information being reported.
- CallLegStartTime (TpDateAndTime)	Date header in INVITE	The time and date when the call leg was started (i.e. the leg was routed).When the SCS received/ sent the SIP INVITE message to initiate the call, if the Date header is not present, the OSA SCS should make a time stamp to be used as this parameter value.
- CallLegConnectedToResourceTime (TpDateAndTime)	N/A	The date and time when the call leg was connected to the resource. If no resource was connected the time is set to an empty string. Either this element is valid or the CallConnectedToAddressTime is valid, depending on whether the report is sent as a result of user interaction.
- **CallLegConnectedToAddressTime** (TpDateAndTime)	ACK message for the INVITE (answer confirmed).	The date and time when the party received the ACK message for the INVITE (answer confirmed). This information may be provided by the SIP server. It tells when the call leg was connected to the destination (i.e. when the destination answered the call). If the destination did not answer, the time is set to an empty string.
- CallLegEndTime (TpDateAndTime)	SIP BYE	Date and time when the call leg was released (e.g. SIP BYE message is sent to participant or received from the participant).
- **ConnectedAddress** (TpAddress)	FROM header URL (OSA terminating call leg) or Request-URI (OSA originating call leg)See Table **6-2: TpAddress** for mapping from SIP	The address of the party associated with the leg. If during the call the connected address was received from the party (SIP Contact header ?) then this is returned, otherwise the destination address (for legs connected to a destination) or the originating address (for legs connected to the origination) is returned
- **CallLegReleaseCause** (TpReleaseCause)	See Table **6-18: TpReleaseCause** for mapping from SIP	The cause of the termination. May be present with P_CALL_LEG_INFO_RELEASE_CAUSE was specified
- **CallAppInfo** (TpCallAppInfoSet)	See Table 6-4: **TpCallAppInfo** for mapping from SIP	Additional information for the leg. May be present with P_CALL_LEG_INFO_APPINFO was specified.
NOTE: A set of TpCallAppInfo.		

5.6.6 getInfoErr

getInfoErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This method is an asynchronous method that reports that the original request was erroneous, or resulted in an error condition.

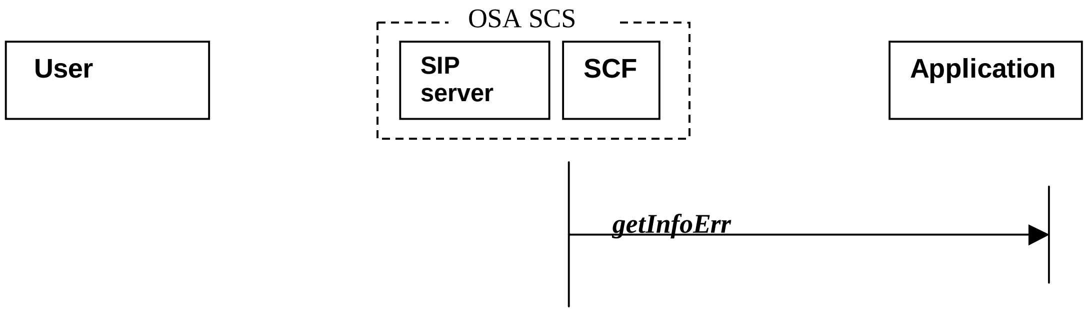

Figure 5-50: Call flow for *getInfoErr()*

Table 5-89: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller, Redirect.**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call leg via the *getInfoReq* method**
1	The original request getInfoReq is erroneous or cannot be accepted due to e.g. call leg terminates abnormally.
2	The SCS identifies the correct applications that requested the call leg information and invokes the ***getInfoErr*** method.

Table 5-90: Parameter mapping

To: *getInfoErr*	**From: SIP**	**Remark**
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError):	See Table 6-5: **TpCallError** for mapping from SIP.	

5.6.7 superviseErr

superviseErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This is an asynchronous method that reports a call leg supervision error to the application.

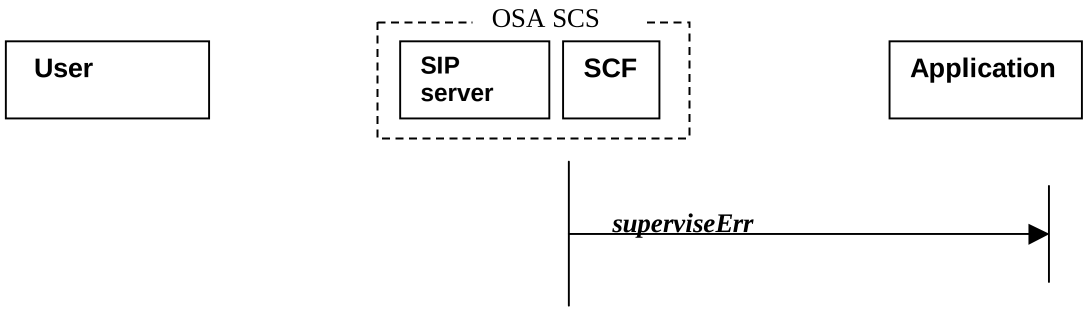

Figure 5-51: Call flow for *superviseErr()*

Table 5-91: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller.** **(Any, except Redirect mode.)** **However, if treatment (TpCallSuperviseTreatment) implies call release, then UA mode of operation is demanded. For this treatment, if the SCS is acting as a proxy, the only SIP message the SCS can generate after receiving superviseRes() in the call leg is BYE.**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call via the *superviseReq* method.**
1	The SCS detects an error that can affect call supervision, e.g. call routing error.
2	The SCS identifies the correct applications that requested the call information and invokes the ***superviseErr*** method.

Table 5-92: Parameter mapping

To: *superviseErr*	**From: SIP 4xx**	**Remark**
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError** mapping from SIP	

5.6.8 superviseRes

superviseRes (callLegSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration): void

This is an asynchronous method that reports a call leg supervision event to the application.

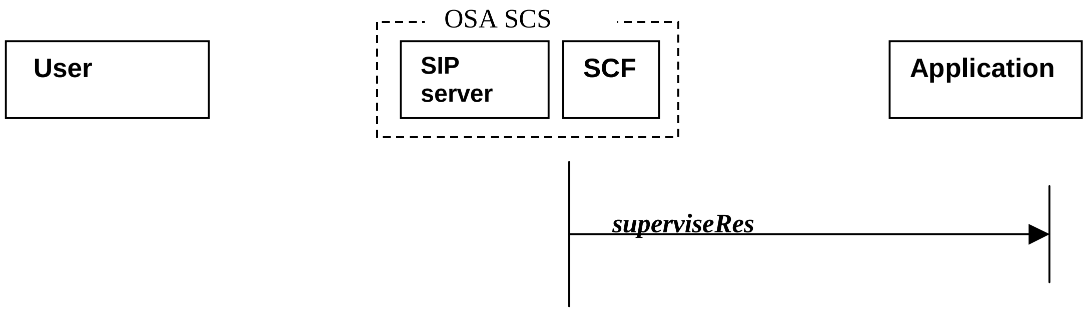

Figure 5-52: Call flow for *superviseRes()*

Table 5-93: Normal operation

SIP Server Mode for the OSA SCS:	**Proxy, UA, B2BUA or 3rd party controller.** **(Any, except Redirect mode.)** **However, if treatment (TpCallSuperviseTreatment) implies call leg release, then UA mode of operation is demanded. For this treatment, if the SCS is acting as a proxy, the only SIP message the SCS can generate after receiving superviseRes() in the call leg is BYE.**
Pre-conditions:	**Call is in progress. The application has requested information associated with a call leg via the** *superviseReq* **method. The specified call leg supervision timer expires.**
1	The SCS detects that the supervision time is expired and acts according to the requested treatment (e.g. release call sending BYE) in ***superviseReq*** . The SCS identifies the correct application and invokes the ***superviseRes*** method.

Table 5-94: Parameter mapping

To: ***superviseRes***	From: SIP 4xx	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-1 to 4-5.	No direct mapping – a correlation.
report (TpCallSuperviseReport)	N/A	Defines the response(s) from the call control service for calls that have been supervised, (e.g. timeout, call-ended, tone-applied, UI-finished).
usedTime (TpDuration)	BYE (release call)	No direct mapping to SIP: TpCallSuperviseTreatment in superviseReq defines the treatment of the call by the call control service when the call supervision timer expires. It may be a request to release (P_CALL_SUPERVISE_RELEASE) the call and /or a request to send a warning tone (P_CALL_SUPERVISE_TONE_APPLIED) to the caller and/or to notify the application The OSA SCS to issue BYE in SIP.
NOTE: The OSA SCS to issue BYE in SIP when the call supervise treatment request is to release the call.		

5.6.9 attachMediaErr

attachMediaErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This asynchronous method reports that the original request was erroneous, or resulted in an error condition.

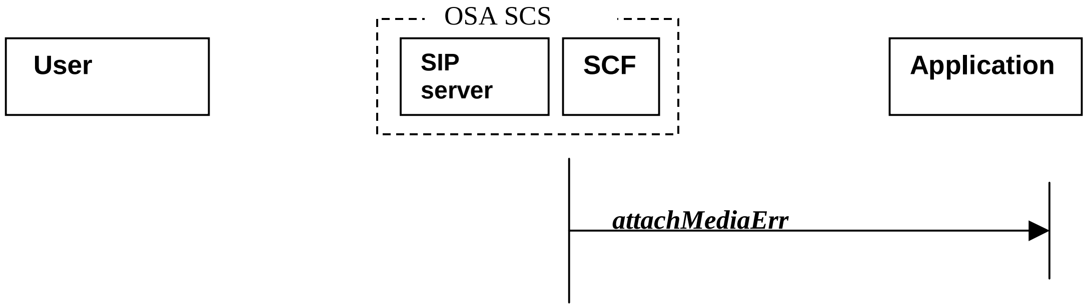

Figure 5-53: Call flow for *attachMediaErr()*

Table 5-95: Normal operation

SIP Server Mode for the OSA SCS:	**UA, B2BUA or 3rd party controller.**
Pre-conditions:	**Call is in progress. The application has requested attach media associated with a call leg via the *attachMediaReq* method.**
1	The SCS detects an error that can affect the call, e.g. call routing error.
2	The SCS identifies the correct applications that requested the attach media and invokes the ***attachMediaErr*** method.
NOTE: A standard User (SIP user agent) should be controllable in the mechanism described here. The mechanism relies on the support of Re-invites by user agent servers.	

Table 5-96: Parameter mapping

To: *superviseErr*	From: SIP 4xx	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError** mapping from SIP	

5.6.10 attachMediaRes

attachMediaRes (callLegSessionID : in TpSessionID) : void

This asynchronous method reports the attachment of a call leg to a call has succeeded. The media channels or bearer connections to this leg are now available.

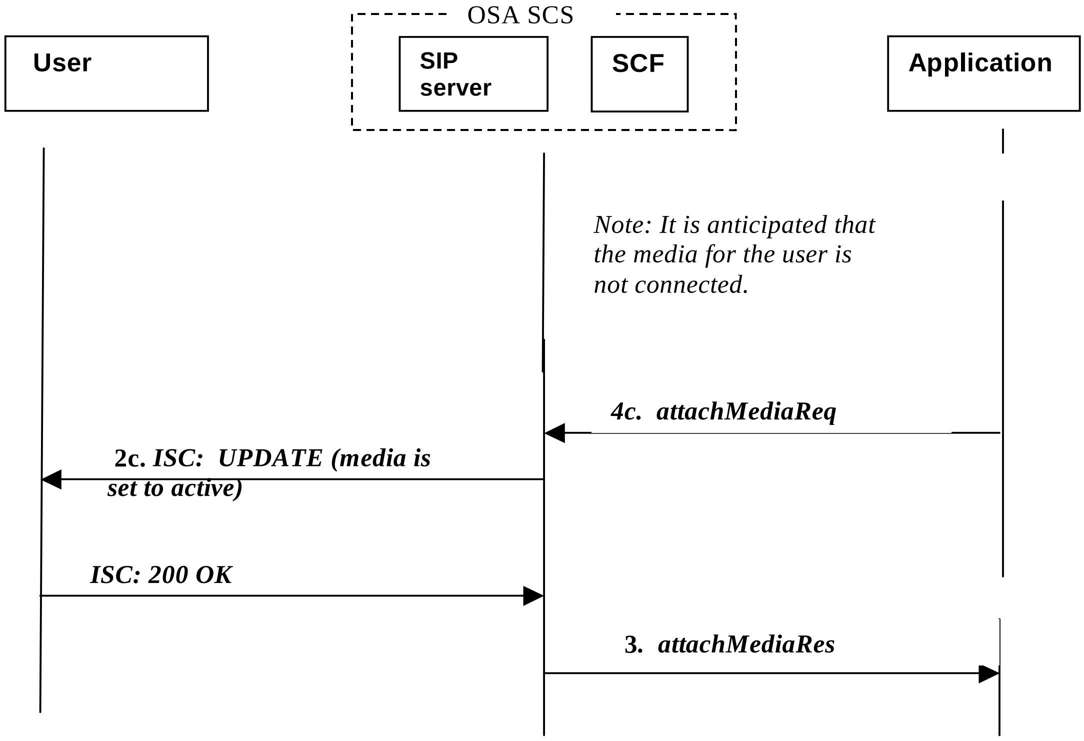

Figure 5-54: Scenario a: Call flow for *attachMediaRes(),* UA/B2BUA mode

Table 5-97: Normal operation

SIP Server Mode for the OSA SCS:	**UA mode** **The generation of a SIP message (UPDATE [12]) on request from the application to attach media channels of a single user in the call demands the SIP server of the OSA SCS to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. The leg is in a connection state and the media communication is on-hold for the call party in its communication with the other legs in the call.** **AttachMedia has bee requested (not executed until the connected state is reached (200 OK /ACK) , i.e. if received before the SCS should buffer the request until it can be executed).**
1	The OSA SCS has requested the media stream(s) for the call leg object to be attached when the call/session state enables this. (The SCS generates a new SIP UPDATE message to be sent toward the user, i.e. in this case the *attachMediaReq()* method is mapped onto a SIP UPDATE message with an SDP on hold.)
2	The OSA SCS confirms the attach media (200 OK /ACK) and notifies the application about the successful attachment of the media stream(s) for the user with the ***attachMediaRes()***
NOTE 1: The media connection is established when application receives the *attachMediaRes()* method. A standard User (SIP user agent) should be controllable in the mechanism described here. The mechanism relies on the support of UPDATE by user agent servers. NOTE 2: See also Annex B and flow example B6.	

Table 5-98: Parameter mapping

From: *attachMediaRes*	To: SIP	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.

5.6.11 detachMediaErr

detachMediaErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void

This asynchronous method reports that the original request was erroneous, or resulted in an error condition.

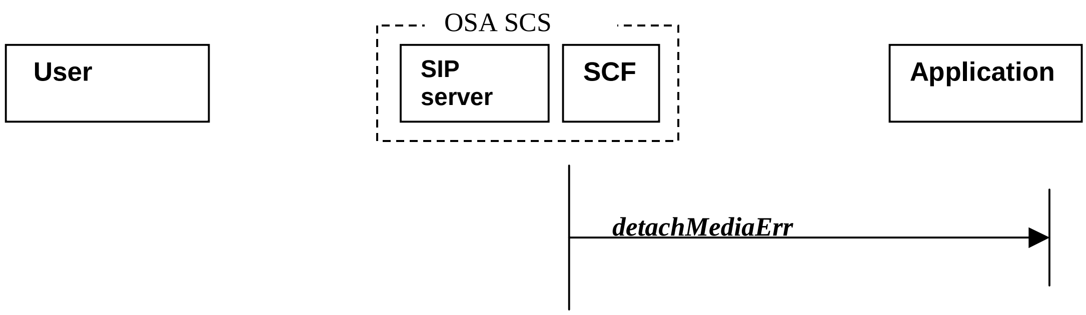

Figure 5-55 Call Flow for *detachMediaErr()*

Table 5-99: Normal operation

SIP Server Mode for the OSA SCS:	**UA, B2BUA or 3rd party controller.**
Pre-conditions:	**Call is in progress. The application has requested detach media associated with a call leg via the *detachMediaReq* method.**
1	The SCS detects an error that can affect the call, e.g. call routing error.
2	The SCS identifies the correct applications that requested the detach media and invokes the ***detachMediaErr*** method.
NOTE: A standard User (SIP user agent) should be controllable in the mechanism described here. The mechanism relies on the support of Re-invites by user agent servers.	

Table 5-100: Parameter mapping

To: *detachMediaErr*	From: SIP 4xx	Remark
callLegSessionID (TpSessionID)	See **"**OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.
errorIndication (TpCallError)	See Table 6-5: **TpCallError** mapping from SIP	

5.6.12 detachMediaRes

detachMediaRes (callLegSessionID : in TpSessionID) : void

This asynchronous method reports the detachment of a call leg from a call has succeeded. The media channels or bearer connections to this leg are no longer available.

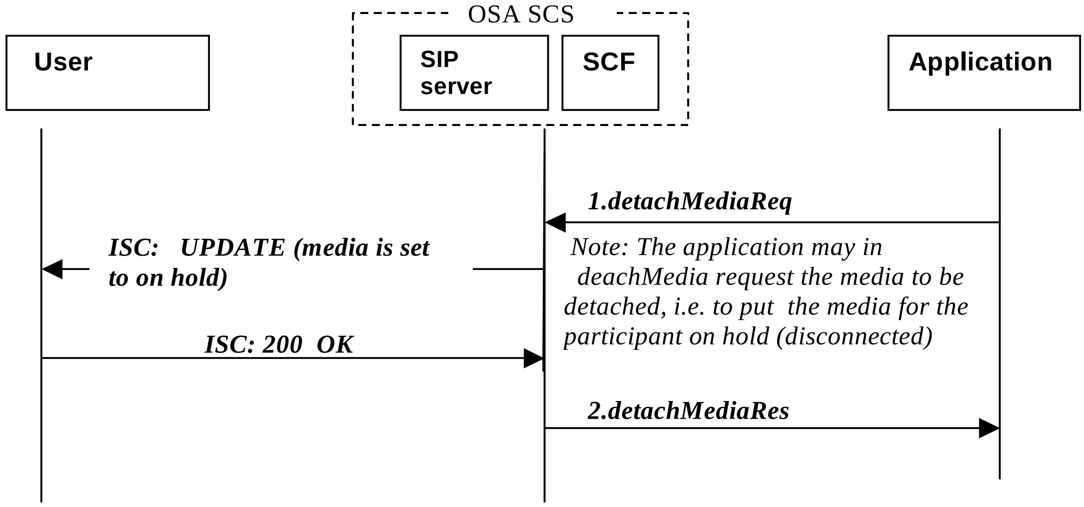

Figure 5-56: Call flow for *detachMediaReq/Res()*, UA/B2BUA mode

Table 5-101: Normal operation

SIP Server Mode for the OSA SCS.	**UA mode** **The generation of a SIP message (UPDATE [12]) on request from the application to detach media channels of a single user in the call demands the SIP server of the OSA SCS to operate in a UA mode (e.g. UAC, B2BUA, 3rd party controller).**
Pre-conditions:	**A relationship between the application and the call including associated call leg object(s) exists. The leg is in a connection state and has a media connection established with the others legs in the call.** **The application has requested to put the media communication on hold for the call party (detach media), by e.g. invoking the *detachMediaReq* method.** **DetachMedia is not executed until the connected state is reached (200 OK /ACK) , i.e. if received before the OSA SCS should buffer the request until it can be executed.**
1	The OSA SCS has requested the SIP server of the OSA SCS to de-attach the media when the call/session state enables this. (The SCS generates a new SIP UPDATE message to be sent toward the user, i.e. in this case the de*tachMediaReq()* method is mapped onto a SIP UPDATE message with an SDP on hold.)
2	The OSA SCS confirms the detach media (200 OK /ACK) and notifies the application about the successful detach media with the ***detachMediaRes()***
NOTE 1: The media on-hold (disconnection) is established when application receives the ***detachMediaRes()*** method. A way to map this functionality in SIP is to use the SDP on hold feature enabling putting the media streams on hold (detach media) while the session is established or after the establishment. A standard User (SIP user agent) should be controllable in the mechanism described here. The mechanism relies on the support of UPDATE by user agent servers. NOTE 2: See also Annex B and flow example B6.	

Table 5-102: Parameter mapping

From: *continueProcessing*	To: SIP	Remark
callLegSessionID (TpSessionID)	See "OSA Call and SIP Dialogue Correlation Tables". Table 4-2 to 4-5.	No direct mapping – a correlation.

6 Detailed parameter mappings

This clause contains detailed parameter mappings for data types that are used in the parameter mapping tables in the previous clauses.

6.1 TpAdditionalCallEventCriteria

Table 6-1:TpAddtionalCallEventCriteria Table mapping

TpAdditionalCallEventCriteria (TpCallEventType)	From SIP (observe for requested additional info)	Remark
Undefined (NULL) (P_CALL_EVENT_UNDEFINED)	N/A	
Undefined (NULL) P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	N/A	
Undefined (NULL) P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORISED	N/A	
MinAddresslength (TpINT32) P_CALL_EVENT_ADDRESS_COLLECTED	N/A	
Undefined (NULL) P_CALL_EVENT_ADDRESS_ANALYSED	N/A	
OriginatingServiceCode (TpCallServiceCode) P_CALL_EVENT_ORIGINATING_SERVICE_CODE	N/A	
OriginatingReleaseCauseSet (TpReleaseCauseSet) P_CALL_EVENT_ORIGINATING_RELEASE	CANCEL or BYE	
Undefined (NULL) P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	N/A	
Undefined (NULL) P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED	N/A	
Undefined (NULL) P_CALL_EVENT_ALERTING	N/A	
Undefined (NULL) P_CALL_EVENT_ANSWER	N/A	
TerminatingReleaseCauseSet (TpReleaseCauseSet) P_CALL_EVENT_TERMINATING_RELEASE	CANCEL, BYE or 4xx, 5xx and 6xx responses	
Undefined (NULL) P_CALL_EVENT_REDIRECTED	N/A	
TerminatingServiceCode (TpCallServiceCode) P_CALL_EVENT_TERMINATING_SERVICE_CODE	N/A	
QueueStatus (TpString)P_CALL_EVENT_QUEUED	SIP 182reason phrase. (See note 1)	Reason phrase is mapped to TpString
NOTE 1: The 182 informational response may be sent several times (e.g. indicating the poison of the calling user in a queue. Furthermore, the message body in the SIP 182 informational response can also be used to carry e.g. music on hold or other media.		

6.2 TpAddress

Table 6-2: TpAddress Table mapping

From: TpAddressRange	To: SIP	Remark
Plan (TpAddressPlan)	SIP	Specifies the address plan in force. Here only all the address schemes which are allowed in SIP are applicable.
AddrString (TpString)	Any URL schemes allowed by RFC 3261	Contains a valid SIP address string. A few examples of SIP URLs: - A user of an online service: "sip:user@xxx.org" "sip:alice@10.1.1.1" - A PSTN phone number at a gateway service: "sip:1212@gateway.com", "sip: +1-212-555-1212:1234@gateway.com; user =phone" An example of tel URL: tel: +1-212-555-1212 Notice: For SIP addresses, wildcards are allowed between the 'sip:' and the '@' in the AddrString, e.g. "sip:*@sales.org" matches all SIP addresses at sales.org:5060.
Name (TpString)	N/A	
Presentation (TpAddressPresentation)	N/A	Defines whether an address can be presented to an end user (presentation allowed or restriced or address not available for presentation) .
Screening (TpAddressScreening)	N/A	Defines whether an address can be presented to an end user. E.g. "user provided address verified and passed" or "Network provided address"
SubAddressString (TpString)	N/A	
NOTE 1: The AddrString defines the actual address information and the structure of the string depends on the Plan. Further information can be found in the OSA API part covering common data definitions [1]. NOTE 2: It should be noted that two SIP addresses will be regarded as equivalent by a gateway if they correspond to the same user at the same network address. The textual form of the two addresses need not be the same. For example, sip:enquiries@yyy.org will be deemed to match <sip:Enquiries@1.2.3.4:5060>Enquiries (if yyy.org resolves to 1.2.3.4).		

6.3 TpAddressRange

Table 6-3: TpAddressRange Table mapping

From: TpAddressRange	To: SIP	Remark
`Plan (TpAddressPlan)`	SIP	Specifies the address plan in force. Here only SIP URL is applicable.
AddrString (TpString)	Any URL schemes allowed by RFC 3261	Contains a valid SIP address string. A few examples of SIP URLs: - A user of an online service: "sip:user@xxx.org" "sip:alice@10.1.1.1" - A PSTN phone number at a gateway service: "sip:1212@gateway.com", "sip: +1-212-555-1212:1234@gateway.com; user =phone" An example of tel URL: tel: +1-212-555-1212 Notice: For SIP addresses, wildcards are allowed between the 'sip:' and the '@' in the AddrString, e.g. "sip:*@sales.org" matches all SIP addresses at sales.org:5060.
Name (TpString)	N/A	
SubAddressString (TpString)	N/A	
NOTE 1: The AddrString defines the actual address information and the structure of the string depends on the Plan. Further information can be found in the OSA API part covering common data definitions [1]. NOTE 2: It should be noted that two SIP addresses will be regarded as equivalent by a gateway if they correspond to the same user at the same network address. The textual form of the two addresses need not be the same. For example, `sip:enquiries@yyy.org` will be deemed to match `<sip:Enquiries@1.2.3.4:5060>Enquiries` (if `yyy.org` resolves to `1.2.3.4`).		

6.4 TpCallAppInfo

Table 6-4: TpCallAppInfo Table mapping

To: TpCallAppInfo	From: SIP	Remark
CallAppAlertingMechanism (TpCallAlertingMechanism)	**Alert-Info**	Indicates the alerting mechanism or pattern to use. When present in an INVITE request, the Alert-Info header field specifies an alternative ring tone to the UAS. When present in a 180 (Ringing) response, the Alert-Info header field specifies an alternative ring back tone to the UAC.
CallAppNetworkAccessType (TpCallNetworkAccessType)	**N/A**	Indicates the network access type (e.g. ISDN) Not mapped. No valid value for SIP in this parameter
CallAppTeleService (TpCallTeleService)	SDP	Indicates the tele service (e.g. telephony). Specifies the type of media indicated in the incoming SDP e.g. data, audio, video, messaging.
CallAppBearerService (TpCallBearerService)	SDP	Indicates the bearer services (e.g. 64kbit/s unrestricted data), this information is carried in SDP under each media type e.g. codec, bandwidth, interleaving….
CallAppPartyCategory (TpCallPartyCategory)	N/A	The category of the calling party. Not mapped. Not defined in SIP
CallAppPresentationAddress (TpAddress)	May be SIP From header field ? This may also be the optional STRING associated to the URI (similar to the name you can associate to an e-mail address)	The address to be presented to other call parties. In case the SIP From header and SIP Contact are different, The From header field may be seen as presentation Address since the UA will only use the contact or via address to decide the routing destination.
CallAppGenericInfo (TpString)	""N/A	Carries unspecified service-service information Service related information transferred over ISC from SCS to S-CSCF is not allowed in the current 3GPP release.
CallAppAdditionalAddress (TpAddress)	N/A	Indicates an additional address. No mapping: Not fined in SIP
CallAppOriginalDestinationAddress (TpAddress)	Request-URI or P-Called-Party-ID	Contains the original address specified by the originating user when launching the call. When the SCS receives an INVITE, if the P-Called-Party-ID header is present, then the SCS uses this header to identify the target address in the resulting outgoing INVITE. If not, then the SCS uses the Request-URI instead.
CallAppRedirectingAddress	N/A	Contains the address of the user from which the call is diverting.

6.5 TpCallError

Table 6-5: TpCallError Table mapping

To TpCallError	From SIP	Remarks
ErrorTime (TpDateAndTime)	N/A	Time should be provided locally by the OSA SCS. Note: In order to have the accurate time, the Timestamp header field may be added to the SIP send by the participant or the SIP server. However, it is not possible to rely on timestamp to be received in message.
ErrorType (TpCallErrorType)	See Table 6-6: **TpCallErrorType** mapping table from SIP	
AdditionnalErrorInfo (TpCallAdditionalErrorInfo)	N/A	See also **TpCallErrorType**

6.6 TpCallErrorType

Table 6-6: TpCallErrorType Table mapping

To: TpCallErrorType	From: SIP	Remark
P_CALL_ERROR_UNDEFINED	*Undefined*	Undefined; the method failed or was refused, but no specific reason can be given.
P_CALL_ERROR_INVALID_STATE	*481 Call/ Transaction Does Not Exist* 491 Request Pending	The call was not in a valid state for the requested operation
P_CALL_ERROR_INVALID_ADDRESS	*404 Not Found,* *413 Request Entity Too Large* *414 Request URI Too Long* *416 Unsupported URI Scheme* *484 Address Incomplete* *485 Ambigous* 488 Not Acceptable Here 604 Does Not Exist Anywhere	The operation failed because an invalid address was given
P_CALL_ERROR_RESOURCE_UNAVAILABLE	*503 Service Unavailable* *606 Not Acceptable*	There are not enough resources to complete the request successfully

6.7 TpCallEventInfo

Table 6-7: TpCallEventInfo Table mapping

To: TpCallEventInfo	From: SIP	Remark
CallEventType (TpCallEventType)	See Table 6-9: **TpCallEventType** mapping from SIP.	
AdditionalCallEventInfo (TpCallAdditionalEventInfo)	See Table 6-9: **TpCallEventType** mapping from SIP.	
CallMonitorMode (TpCallMonitorMode)	See Table 6-13: **TpCallMonitorMode** mapping from SIP.	
CallEventTime (TpDateAndTime)	N/A	Timestamp provided by OSA SCS at event reporting.

6.8 TpCallEventRequest

Table 6-8: TpCallEventRequest Table mapping

To TpCallEventRequest	From SIP	Remark
CallEventType (TpCallEventType)	See Table 6-9: **TpCallEventType** mapping from SIP	.
AdditionalCallEventCriteria **(TpAdditionalCallEventCriteria)**	See Table 6-1: **TpAdditionalCallEventCriteria** mapping from SIP	
CallMonitorMode **(TpCallMonitorMode)**	See Table 6-13: **TpCallMonitorMode** mapping from SIP	

6.9 TpCallEventType

Table 6-9: TpCallEventType Table mapping

To TpCallEventType	From SIP	Remark
P_CALL_EVENT_UNDEFINED	N/A	No mapping from SIP.
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	INVITE	Originating Call Leg event. Not applicable to SIP; would mean an empty To: header.
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORISED	INVITE	Originating Call Leg event.
P_CALL_EVENT_ADDRESS_COLLECTED	INVITE	Originating Call Leg event. No direct mapping to any SIP Method/Response. Correspond to the point in processing where INVITE is received and no location service lookup performed yet, i.e. before destination address determined.
P_CALL_EVENT_ADDRESS_ANALYSED	INVITE	Originating Call Leg event. No direct mapping to any SIP Method/Response. Correspond to the point in processing where INVITE is received and destination address is determined after location service lookup has been performed.
P_CALL_EVENT_ORIGINATING_SERVICE_CODE	INVITE	Originating Call Leg event. RE-INVITE case - mapping ffs.
P_CALL_EVENT_ORIGINATING_RELEASE	BYE, CANCEL See corresponding Table for details	Originating Call Leg event. Request for termination of session from calling party.
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	INVITE	Terminating Call Leg event. Incoming INVITE received at destination requesting the termination of the session (i.e. dialogue invitation request) for callee.
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED	INVITE	Terminating Call Leg event. Incoming INVITE received at destination requesting the establishment of the terminating session for the callee
P_CALL_EVENT_ALERTING	SIP: 180 Ringing	Terminating Call Leg event. The user agent receiving the INVITE is trying to alert the callee. This response may be used to initiate local ring-back for the caller. Note: Implies that the corresponding INVITE request passed through the OSA SCS
P_CALL_EVENT_ANSWER	200 OK for INVITE	Terminating or Originating Call Leg event. A 200 OK for INVITE means the call is answered by called user. Note: Implies that the corresponding INVITE request passed through the OSA SCS.

P_CALL_EVENT_TERMINATING_RELEASE	BYE, 4xx, 5xx, 6xx See corresponding Table for details	Terminating Call Leg event. Request for termination of session (i.e. release of dialogue) from called party/destination.
P_CALL_EVENT_REDIRECTED	3xx responses	Terminating Call Leg event. This status codes are used to indicate that the call is being redirected to a different (set of) destination(s). The redirection address contained in the responseContact header in the 3xx response is to be reported in the **CALL_EVENT_REDIRECTED** event (ForwardAddress field additional event info) to the application.
P_CALL_EVENT_TERMINATING_SERVICE_CODE	N/A	Terminating Call Leg event.
P_CALL_EVENT_QUEUED	SIP:182 Queued	Terminating Call Leg event. In case of ISC, implies that the corresponding INVITE request passed through the OSA SCS.

6.10 TpCallInfoType

Table 6-10: TpCallInfoType Table mapping

From: TpCallInfoType	From: SIP	Remark
P_CALL_INFO_UNDEFINED	N/A	-Undefined
P_CALL_INFO_TIMES	N/A	- Relevant call time
P_CALL_INFO_RELEASE_CAUSE	See Table 6-17, 6-18: **TpReleaseCause** for mapping from / to SIP	- Call release cause
P_CALL_INFO_INTERMEDIATE	N/A	- Send only intermediate reports. When this is not specified the information report will only be sent to the application when the call has ended. When intermediate reports are requested a report will be sent between follow-on calls, i.e. when a party leaves the call.
NOTE: Defines the type of call information requested and reported. The values may be combined (logical 'OR').		

6.11 TpCallLegInfoType

Table 6-11: TpCallLegInfoType Table mapping

From: TpCallLegInfoType	From: SIP	Remark
P_CALL_LEG_INFO_UNDEFINED	N/A	Undefined
P_CALL_LEG_INFO_TIMES	N/A	Relevant call times
P_CALL_LEG_INFO_RELEASE_CAUSE	See Table 6-17	Call leg release cause
P_CALL_LEG_INFO_ADDRESS	See Table 6-2	Call leg connected address.
P_CALL_LEG_INFO_APPINFO	N/A	Call leg application related information
NOTE: Defines the type of call leg information requested and reported. The values may be combined by a logical 'OR' function.		

6.12 TpCallLegConnectionProperties

Table 6-12: TpCallLegConnectiomProperties Table mapping

From: TpCallLegConnectionProperties	To: SIP	Remark
P_CALLLEG_ATTACH_IMPLICITLY	N/A	SIP 200 OK message directly sent. It means that the callLeg should be implicitly attached to the call. In this case, the mapping to SIP is done naturally since in SIP, the natural behaviour is to start media session with others parties in the call once the signalling is established (INVITE, 200 OK, ACK)
P_CALLLEG_ATTACH_EXPLICITLY	Putting media stream in SDP inactive.	It means that the callLeg should be explicitly attached to the call. In this case, the mapping to SIP is done so as to start media session with putting the media stream inactiveonce the dialog is established (INVITE with SDP "on hold", 200 OK, ACK) Attach method need to be called by the application to establish the media connection. See description for attachMedia().

6.13 TpCallMonitorMode

Table 6-13: TpCallMonitorMode Table mapping

From TpCallMonitorMode	To SIP	Remarks
P_CALL_MONITOR_MODE_INTERRUPT	N/A Processing interrupted	SIP Server set to observe for SIP event as requested and if encountered interrupt SIP processing, notify the application and await a request to resume processing.
P_CALL_MONITOR_MODE_NOTIFY	N/A Processing Notify And Continue	SIP server set to observe for SIP event as requested and if encountered notify the application.; SIP Processing continues.
P_CALL_MONITOR_MODE_DO_NOT_MONITOR	N/A Processing transparent	SIP server set not to observe for SIP event –no application interest. It implies there is no initial filtering for the associated indicated event

6.14 TpCallNotificationReportScope

Table 6-14: TpCallNotificationReportScope Table mapping

To: TpCallNotificationReportScope	From SIP	Remark
DestinationAddress (TpAddressRange) If transaction issued from caller (e.g. INVITE) OR OriginatingAddress, if transaction from callee (e.g. Re-INVITE, BYE)	SIP Request-URI header field for originating case or P-Called-Party-ID header for terminating case	UEs can put anything into From and To header which is untrustful, so From and To header can not be used to identify the originating address or destination address.'
OriginatingAddress (TpAddressRange) If transaction from caller (e.g. INVITE) OR DestinationAddress , if transaction issued from caller (e.g. Re-INVITE, BYE)	SIP From header field URL	Depends on applied filtering criteria
NotificationCallType (TpNotificationCallType)	N/A	Indicates if the notification was reported

6.15 TpCallNotifiationRequest

Table 6-15: TpCallNotificationRequest Table mapping

From: TpCallLegInfoType	To: SIP	Remark
CallNotificationScope (TpCallNotificationScope):		
DestinationAddress (TpAddressRange)	URL schemes allowed in RFC 3261 (see NOTE)	Parameter specific to filtering criteria (event triggering) of destination address information. Address plan that can only be accepted are SIP URLs or tel URLs.
OriginatingAddress (TpAddressRange)	SIP URL (see NOTE)	Parameter specific to filtering criteria (event triggering) of originating address information (like e.g. in From header Field in SIP messaging). Address plan can be any, which is allowed in RFC 3261.
CallEventsRequested (set): (TpCallEventsRequest (set) Note: A set of TpCallEventRequest	See Table 6-8: TpCallEventRequest mapping from SIP	
NOTE: The SIP server responsible for event filtering (e.g. S-CSCF) is to monitor for SIP events requested to be notified if encountered to the application.		

6.16 TpCallTreatmentType

Table 6-16: TpCallTreatmentType mapping

TpCallTreatmentType	To SIP	Remark
P_CALL_TREATMENT_DEFAULT	undefined	Depends on any applied default
P_CALL_TREATMENT_RELEASE	SIP: 503 Service Unavailable	Service Unavailable response sent to deny invite request for a new session .Already established call sessions are not affected
P_CALL_TREATMENT_SIAR	SIP: 503 Service Unavailable or BYE	BYE only after user interaction if it implies and established session (e.g. to MRF) Service Unavailable response sent to deny invite request for a new session.
NOTE: Already established call sessions should not be affected by the overload call treatment.		

6.17 TpReleaseCause, mapping to SIP response

Table 6-17: TpReleaseCause Table mapping to SIP

From: TpReleaseCause	To: SIP	Remark
P_UNDEFINED	N/A See Note 3	
P_USER_NOT_ AVAILABLE	480 Temporarily Unavailable	The callee is currently unavailable. Normal call clearing, unspecified reason. Note: No support for inclusion of additional information in the Retry-After header. This header in the response may indicate a better time to call.
P_BUSY	486 Busy Here	The callee is currently not willing or able to take additional calls (user busy). Note: No support for include additional information in the Retry-After header. This header in the response may indicate a better time to call.
P_NO_ANSWER	603 Decline	The callee explicitly does not wish to or cannot participate in the call. Note: No support for include additional information in the Retry-After header. This header in the response may indicate a better time to call.
P_NOT_REACHABLE	480 Temporarily Unavailable	The callee is currently unavailable. The user is absent or not reachable e.g. MS turned off or out of coverage area.
P_ROUTING_FAILURE	404 Not Found	The user does not exist at the domain specified in the Request-URI. This status is also returned if the domain in the Request-URI does not match any of the domains handled by the recipient of the request.
P_PREMATURE_DISCONNECT	N/A See Note 3	
P_DISCONNECTED	N/A See Note2. See Note 3	Normal call clearing. Recommended value when an established session is to be released.
P_CALL_RESTRICTED	403 Forbidden	
P_UNAVAILABLE_RESOURCE	503 Service Unavailable	
P_GENERAL_FAILURE	500 Server Internal Error	
P_TIMER_EXPIRY	408 Request Timeout	
NOTE 1: SIP CANCEL will be sent if any pending invitations (INVITE) to be cancelled in response to the release() method independent of TpReleaseCause value NOTE 2: SIP BYE will be sent if an established session (SIP leg) is to be released in response to the release() method independent of TpReleaseCause value. However, the recommended value is in this case P_DISCONNECTED. NOTE 3: Where no mapping is defined, a default mapping to 480 Temporarily Unavailable is recommended.		

6.18 TpReleaseCause, mapping from SIP

Table 6-18: TpReleaseCause Table mapping

From: TpReleaseCause	To: SIP	Remark
P_UNDEFINED	N/A	No mapping
P_USER_NOT_AVAILBLE	404 Not Found 410 Gone 604 Does Not Exist Anywhere	The callee is unavailable. e.g. the address of callee might have been changed.
P_BUSY	486 Busy Here 600 Busy EveryWhere	The callee is not able or not willing to accept additional call
P_NO_ANSWER	603 Decline	The callee explicitly does not wish to or cannot participate in the call.
P_NOT_REACHABLE	480 Temporarily Unavailable	User is not logged in or user's terminal is out of radio coverage.
P_ROUTING_FAILURE	400 Bad Request, 420 Bad Extension, 482 Loop Detected, 483 Too Many Hops 484 Address Incomplete 485 Ambiguous,	
P_PREMATURE_DISCONNECT	SIP CANCEL 480 Temporarily Unavailable	Pending invitation (INVITE) abandoned by caller before answer (i.e. before the request has been acknowledged (ACK)) or user's terminal is out of radio coverage.
P_DISCONNECTED	SIP BYE	Normal call clearing
P_CALL_RESTRICTED	403 Forbidden	
P_UNAVAILABLE_RESOURCE	503 Service Unavailable	
P_GENERAL_FAILURE	500 Server Internal Error, 501 Not Implemented, *502 Bad Gateway,* 505 Version Not Supported	
P_TIMER_EXPIRY	408 Request Timeout, 504 Gateway Timeout	

6.19 TpAoCInfo

Table 6-19: TpAoCInfo Table mapping

From: TpAoCOrder	To: SIP	Remark
ChargeOrder (TpAoCOrder)	See Table 6-20: **TpAocOrder**	
Currency (TpString)	N/A	Currency unit according to ISO-4217:1995
NOTE: Defines the Sequence of Data Elements that specify the Advice Of Charge information to be sent to the terminal.		

6.20 TpAoCOrder

Table 6-20: TpAoCOrder Table mapping

From: TpAoCOrder	To: SIP	Remark
TpAoCOrderCategory:	-	
P_CHARGE_ADVICE_INFO (TpChargeAdviceInfo)	N/A	
P_CHARGE_PER_TIME (TpChargePerTime)	N/A	
P_CHARGE_NETWORK (TpString)	N/A	
NOTE: In the current 3GPP release, how to transmit AoC information to UE using ISC is not addressed, it maybe addressed in future release.		

Annex A:
Introduction to API Mapping for OSA MPCCS

A.1 OSA Service Provision for MPCCS in IMS

The figure below depicts an overall view of how MPCC services can be provided.

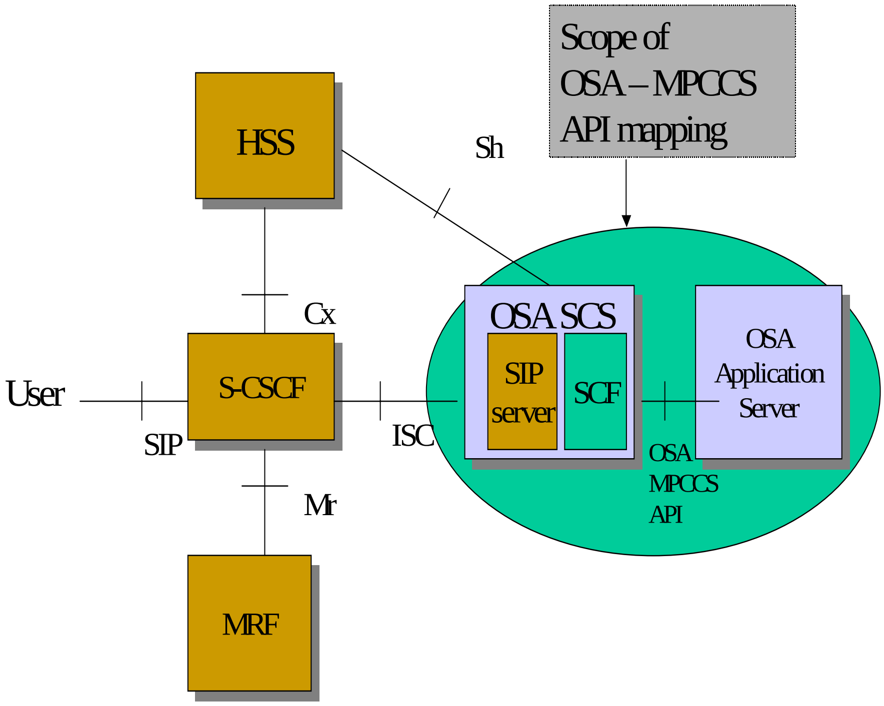

Figure A-1: Functional architecture for support of MPCCS Service Provision for IP Multimedia subsystem

The OSA Service Capability Server (OSA SCS) is the "controlling entity" and the Serving-Call Session Control Function (S-CSCF) is the "controlled entity" .The MRF is the Media Resource Function. (MRF).

ISC: This reference point is the Internal Service Control Interface, used between the S- CSCF and the OSA SCS. The ISC interface is based on Session Initiation Protocol (SIP), which is specified in 3GPP TS 24.229[12].

Cx: The Cx reference point supports information transfer between CSCF and HSS.
The protocol used between the S-CSCF and HSS (Cx Interface) is specified in 3GPP TS 29.228[8].

Sh: The Sh reference point supports information transfer between OSA SCS and HSS.
The protocol used between the OSA SCS and HSS (Sh Interface) is defined in 3GPP TS 29.328 [15].

Mr: This reference point allows interaction between an S-CSCF and an MRF (i.e. the Media Resource Function controller, MRFC). The protocol used for the Mr reference point is based on SIP, which is specified in 3GPP TS 24.229[12].

Filtering is done in the S-CSCF on SIP initial request messages only. It can e.g. be based upon:

- Any initial known or unknown SIP method (e.g. REGISTER, INVITE, SUBSCRIBE, MESSAGE);

- Direction of the request is with respect to the served user – either mobile originated (MO) or mobile terminated (MT) to registered user; or mobile terminated to unregistered user;

- Session description information;

- The present/absent content of a particular SIP header.

Filter Criteria (FC) is the information the S-CSCF receives from the HSS that defines the criteria based on which the S-CSCF shall send the SIP initial request to the OSA SCS. Then the application can decide whether to be in the path of all the subsequent SIP messages of this dialog or not. For more detail on initial filter criteria and triggering mechanisms in the S-CSCF, see 3GPP TS 23.218 [6].

Initial Filter Criteria (iFC) are filter criteria that are stored in the HSS as part of the user profile and are downloaded together with addresses of the assigned application servers (e.g., OSA SCS addresses) via the Cx interface to the S-CSCF upon user registration or upon a terminating initial request for an unregistered user if unavailable. They represent a provisioned subscription of a user to an application. Application server specific data is also exchanged between HSS and the OSA SCS during registration via Sh interface.
After downloading the User Profile from the HSS, the S-CSCF accesses the filter criteria. Initial Filter Criteria are valid throughout the registration lifetime of a user or until the User Profile is changed.

A.2 MPCCS

A.2.1 Introduction

The MPCCS allows an application to establish multi-party calls where several legs can simultaneously be connected.. In fact, the MPCCS as defined, allows application to create a leg and to route it. In SIP, to establish a session it requires at least two SIP endpoints (UAs).
MPCCS which beside 2-party call encompasses application initiated 1 party and multi-party calls can be mapped to SIP implying the OSA SCS behaves as a SIP application server on the ISC interface.

A.2.2 SIP Server Roles in OSA SCS

A.2.2.1 Introduction

The OSA SCS behaves as a SIP server toward the ISC interface.
The SIP application server hereby may act in different roles or modes The role of UAC and UAS as well as proxy and redirect servers are defined on a transaction-by-transaction basis.
For example, the user agent initiating a call acts as a UAC when sending the initial INVITE request and as a UAS when receiving a BYE request from the callee.

Similarly, the same software can act as a proxy server for one request and as a redirect server for the next request.

However, besides these modes of operation for more advanced service application demands also the Back-to-Back User Agent (B2BUA) and 3rd Party controller modes have been defined.

The OSA SCS possible different modes of SIP server operation is described in the following.

A.2.2.2 OSA SCS acting as a SIP Proxy server

In this mode of operation the incoming SIP Request is proxied by the S-CSCF to the OSA SCS, which then acts as a SIP proxy server proxying the Request back to the S-CSCF which then proxies it towards the destination.

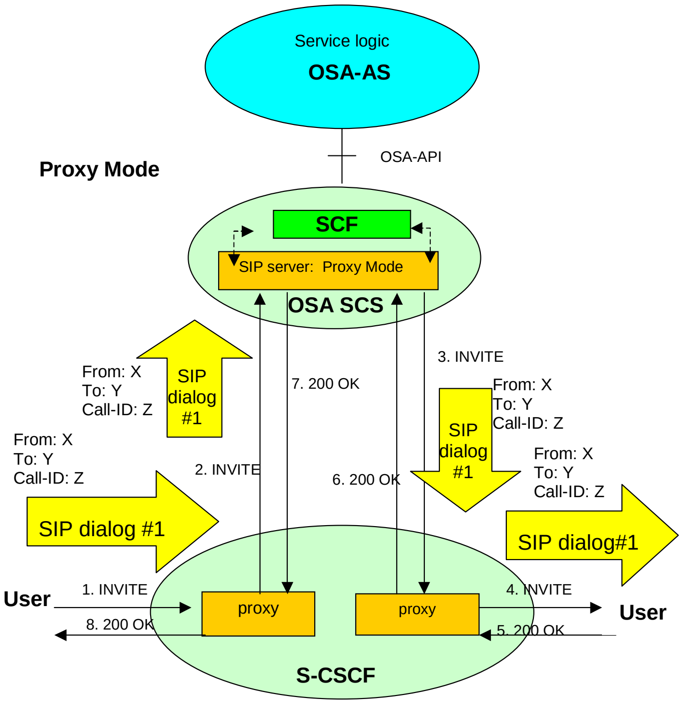

Figure A-2: Example OSA SCS Proxy Server Mode operation

- Scope:

Service applications that need to manipulate data conveyed in the SIP signalling between a UAC and a UAS, like changing destination address (call forwarding services), but do not demand to intervene on the call as such.

During the proxy operation the OSA SCS may add, remove or modify the header contents contained in the SIP request according to the Proxy rules specified in [14].
Applicable for 2-party calls. However, forking may occur resulting in more SIP dialogues being established between the Caller) UAC and 2 or more callees (UASs).

- Constrains:

The control and visibility of forking in the application is not currently covered by the OSA API MPCCS.

A.2.2.3 OSA SCS acting as Redirect server

In this mode of operation the incoming SIP Request is proxied by the S-CSCF to the OSA SCS which then acts as a Redirect Server as specified in [14].

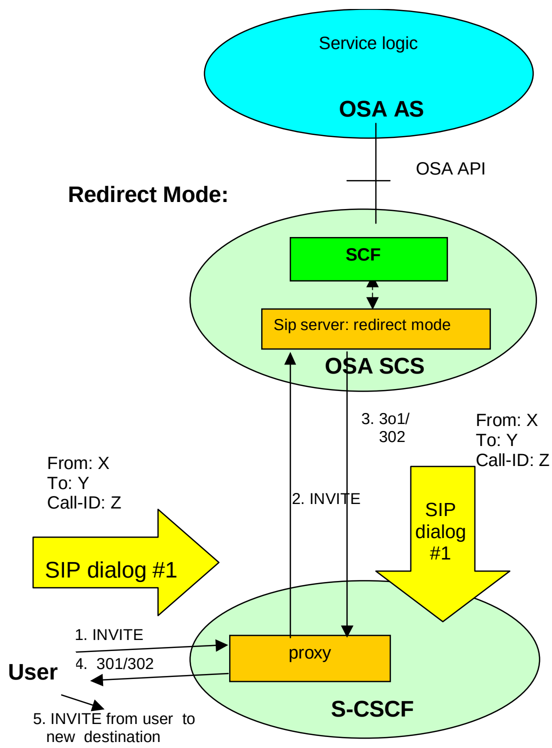

Figure A-3: Example OSA SCS Redirect Server Mode operation

- Scope:

Service applications that need to request a redirection of a call by the network to a new destination, e.g. due to number changed (callee moved). Hereby the application is to provide the new contact address(es) and leave the call.

During the Redirect operation the OSA SCS may terminate the dialog by requesting a call redirection given a list of 1 or more possible new addresses to contact contained in the redirection response request according to the Redirect rules specified in [14].

- Constrains:

NOTE: The control and possibility of requesting a redirection (3xx response) is not currently supported by the OSA MPCCS API.

A.2.2.4 OSA SCS acting as UA

- SIP User Agent Terminating (UAt)
  In this mode of operation the incoming SIP Request is proxied by the S-CSCF to the OSA SCS which then acts as a terminating UA (UAS) as specified in [14].

- SIP User Agent Originating (UAo)
  In this mode of operation the OSA SCS acts as an originating UA (UAC) as specified in [14] and generates a SIP Request which it sends to the S-CSCF which then proxies it towards the destination.

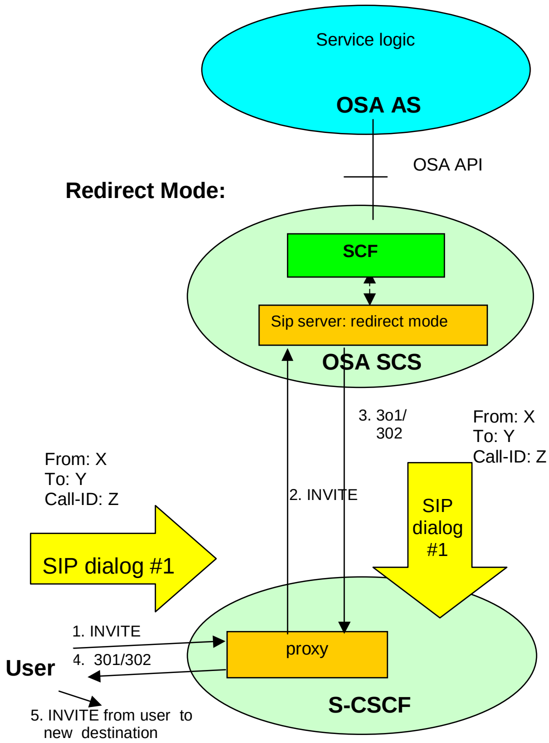

Figure A-4: Example OSA SCS User Agent Server Mode operation

- Constrains:

NOTE: Any direct control of media resources by the OSA SCS when acting as UA is outside the scope of this specification.

A.2.2.5 OSA SCS acting as a B2BUA

In this case the controller, i.e. the OSA SCS, takes over the ownership of the call set-up by a different party by acting as a Back-to-Back User Agent (B2BUA). The OSA SCS looks deceptively like a proxy, but it is not. The OSA SCS acts as a UAS for the INVITE received from caller (UAC), and then as a UAC when it initiates a call to the callee (UAS).

In this case the incoming SIP Request is proxied by the S-CSCF to the OSA SCS which then generates a new SIP Request for a different SIP dialog which it sends to the S-CSCF which then proxies it towards the destination.
In this mode the OSA SCS behaves as a B2BUA for the multiple SIP dialogs as specified in [14].

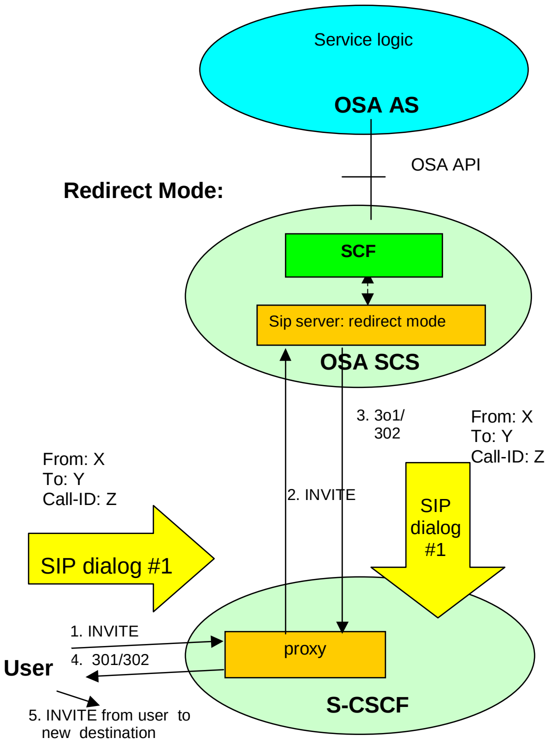

Figure A-5: Example OSA SCS B2BUA Server Mode operation

- Usage:

Service applications that need advanced signalling control, i.e. the capability to intervene on a call.
Some examples may be applications that needs to release a call (e.g. prepaid service) or a single user, or add or replace a user (follow-on call), or needs to generate messages during the call or act on mid-call events from a call party (e.g. re-INVITE).

EXAMPLE: Pre-Paid card service runs out of money: the application may generate some message to the user and/or release the user.

- Constrains:

The mode B2BUA is to be determined based on SIP requests messages. It is not allowed in this release that a proxy can change to a B2BUA in the middle of a dialog, unless the purpose of doing this is to release a dialog. Where it cannot be known in advance if the application demands Proxy mode or B2BUA mode, the default should for the OSA SCS be to act as a B2BUA.

NOTE: Notice that the end-to-end call (SIP dialogue) between caller and callee will become divided t into a multitude of different "end-to-end" calls (SIP dialogues) , where the B2BUA concept is applied.

A.2.2.6 OSA SCS acting as a 3rd Party Controller

In this mode the OSA SCS generates a new SIP Request for a different SIP dialog and sends it to the S-CSCF which then proxies it towards the destination. The OSA SCS may generate one or more different SIP dialogues in this way. This may be combined with the OSA SCS behaviour as a B2BUA for the multiple SIP dialogs as specified in RFC3261 [14], i.e. when more than 2 parties are involved in the call.

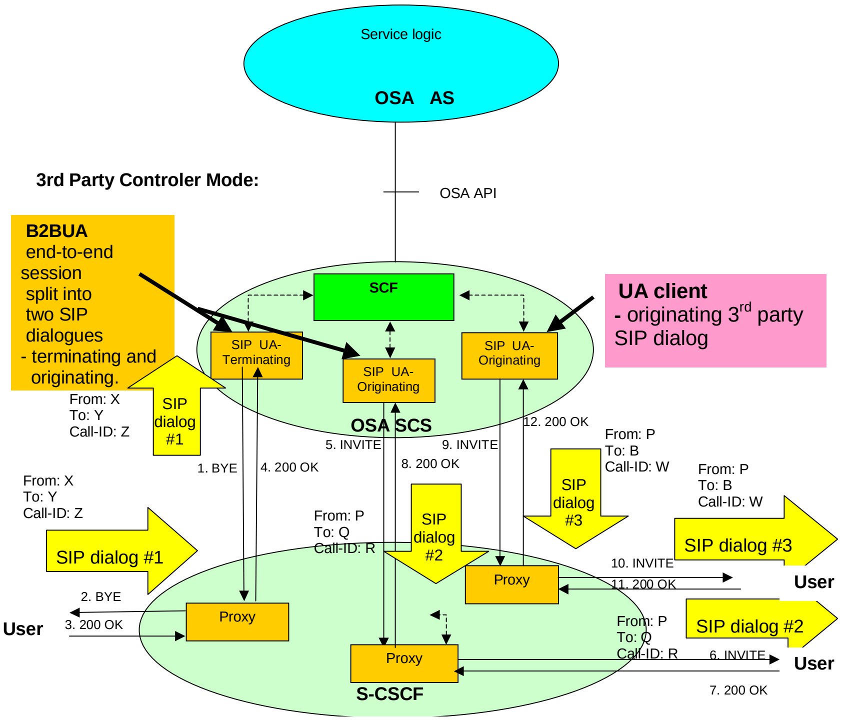

Figure A-6: Example OSA SCS 3rd Party Controller Server Mode operation

- Usage:

Application initiated one party , two-party and multi-party calls.
It may also be associated with B2BUA mode of operation, e.g. where the application demands to invite a 3rd part into a 2-party.

- Constrains:

The control of media resources for application initiated calls is outside the scope of this specification.

A.2.3 SIP Server Role Mode Transitions

Figure 5 provides an overview of the states and transitions of the FSM for Call Control Signalling Terminations. These states and transitions are more precisely defined in the following clauses.

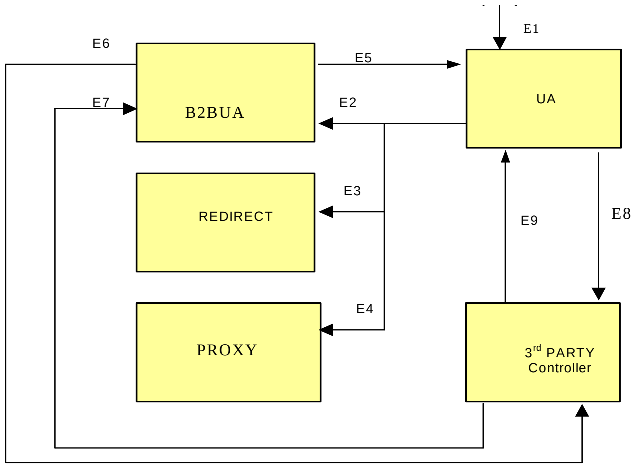

Figure A-7: Operation Mode for the OSA SCS

The server mode diagram above for the OSA SCS shows the possible mode transitions. It contains the following transitions (events):

E1	Incoming Invite received from the network (caller) or request received from the application to initiate a call "out of the blue". detected
E2	Application request to act as B2BUA on call received from the network
E3	Application request to act as Redirect server on call received from the network
E4	Application request to act as Proxy server on call received from the network
E5	Application request to act as single UA on call received from the network
E6	Application request to act as 3rd Party controller on call received from the network
E7	Application request to act as B2BUA on call received from the network
E8	Application request to act as 3rd Party controller on call initiated from application
E7	Application request to act as single UA.

Annex B: SDP in SIP at application controlled calls for OSA MPCCS API

B.1 Introduction

A mechanism is needed that allows a controller like OSA SCS to create, modify, and terminate calls with other entities.. Third party call control refers to the ability of one entity, in this case the OSA SCS to create a call in which communications are actually between other parties. A SIP mechanism for accomplishing third party call control that does not require any extensions or changes to SIP is presented. It is merely an application of the tools enabled through the SIP specification RFC 3261 [14]. It enables a controller like the OSA SCS to create calls/sessions with any entity that contains a normal SIP User Agent. Annex B is based upon the principles described in "Third Party Call Control in SIP" [16].

B.2 OSA SCS and Application based Call and Media Control

Third party call control is a set of good design patterns for how to implement a service that needs to be in control of a session. The B2BUA mechanism is just one pattern that the 3rd party call controller can use to get control of a session. A B2BUA is a mechanism that allows a controller to take over the control of a session initiated by another party. Once in control it can control the session by generating requests and responses on the different call-legs. OSA SCS can of course also at all times initiate a session or a new transaction within a given SIP dialogue hereby acting as a User Agent or 3rd party call controller.

The basic principle behind the third party mechanism applied for OSA MPCCS application initiated calls is simple. The OSA SCS acting as a controller on request from the OSA application first calls one of the users, A, and presents the INVITE without any media. When this call is complete, the OSA SCS has the SDP needed to communicate with user A. The OSA SCS can then, if so requested by the OSA application, use SDP A to establish a call to user B. When this call is completed, the OSA SCS has the SDP needed to communicate with user B. This information is then passed to user A. The result is that there is on request from the application established an OSA call leg (SIP dialogue) between the OSA SCS and user A, and a call leg (SIP dialogue) between the OSA SCS and user B, but media between user A and user B.

The aim here is to keep the OSA application based session control for MPCCS as simple as possible, but also generally useable, and avoid SDP awareness in the OSA SCS acting as the controller..

In the following some example scenarios for illustrating a possible handling of SDP in SIP at OSA MPCCS application controlled call sessions are given.

NOTE 1: A user may herein be presented by any entity that contains a normal SIP User Agent. For example a user could be represented by an ordinary call party (e.g. SIP enabled phone/PC), a gateway or a network entity like e.g. a Conference Server or MRF.

NOTE 2: Where an OSA application demands to control (e.g. restrict call to a given media type (e.g. voice),) which media types should be allowed on a call, it can also use the Multimedia Call Control Service (MMCCS), which enhances the MPCCS with multimedia control capabilities (allows e.g. the application to bar certain media type(s)).

B.3 Example OSA SCS Application initiated One-Party Call

An example of an application initiated One-Party Call could be a booked "wake-up call" or "reminder call", i.e. a call that is to be set-up at a predefined time and date from the network initiated by an OSA application using the MPCCS.

The recommended flow is as follows: The application requests a call to be set-up to user A.
The OSA SCS sends an INVITE to the user A, without any SDP (it means that the OSA SCS does not need to assume anything about the media of the devices). User A responds with its SDP a1, in a 200 OK, which is immediately ACK'ed with an on-hold SDP generated by the OSA SCS.

A flow example for a One Party call set-up from application is illustrated in the figure below:

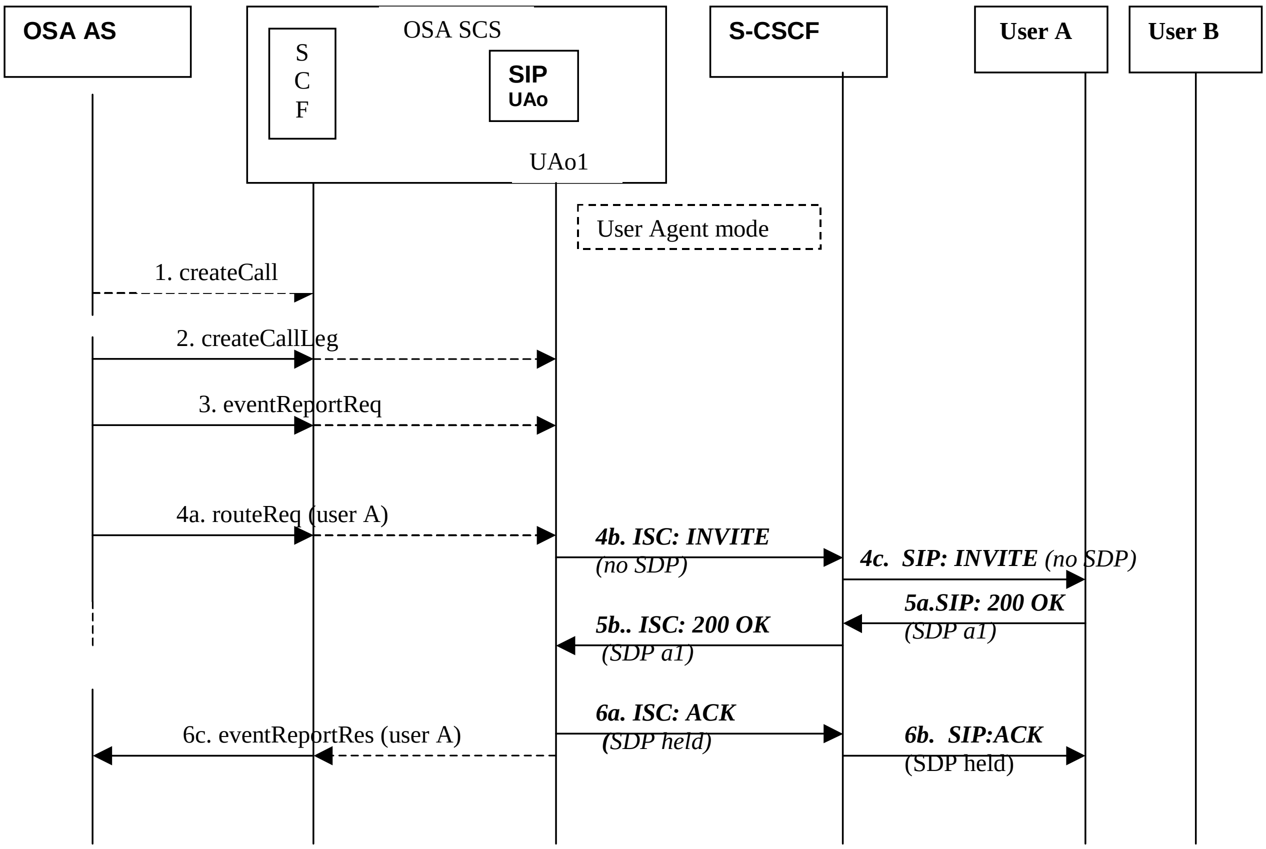

Figure B-1 Example Initiating OSA SCS Flow for One Party call Set-up

A description for the flow is given below:

1: This message requests the OSA SCS to create a call object (an object implementing the IpMultiPartyCall interface). Assuming that the criteria for creating a call object implementing the IpMultiPartyCall interface (e.g. load control values not exceeded) is met it is created.

2: This message instructs the OSA SCS to create a call leg (the object implementing the IpMultiPartyCall interface) for user A.

3: This message requests the call leg for user A to inform the application when the call leg answers the call.

4a: The created OSA terminating call leg is requested to route the call/session to the specified destination for user A.

4b: The OSA SCS acting as a logical UAo1 generates an INVITE request message with no SDP on the ISC interface to S-CSCF providing the destination address of user A.
The OSA SCS SIP server is in SIP UA Originating Endpoint mode.

4c: The S-CSCF proxies the INVITE request toward user A.

5a: User A answers the call and responds with its SDP (SIP 200 OK including SDP a1)
Note: It is here only shown that the call is answered by user A, e.g. user A accepting the incoming call and sending a 180(Ringing) back to the UAo1 on OSA SCS is omitted for simplicity reasons !..

5b: The S-CSCF proxies the SIP 200 OK including SDP a1 to the originating UAo1 in the OSA SCS via the ISC interface.

6a: The OSA SCS being the controller immediately generates an ACK with an on-hold SDP being send on the ISC interface to the S-CSCF. It hereby takes SDP a1, and generates another SDP which has the same media composition, but is on hold.

6b: The S-CSCF proxies the ACK with SDP on hold toward user A.

6c: The leg object (implementing user A's IpCallLeg interface) in OSA SCS passes the result of the call being answered back to the application in OSA AS.

General Remarks:

The OSA SCS operation in User Agent mode provides a central point for signalling control, as the application hereby is offered complete control over the call.

B.4 Example OSA SCS Application initiated Two-Party Call

An example of an application initiated Two-Party Call could be a Click-to dial service, that allows a user to click on a web page when wished to speak to a customer service representative. The web-server then via some "stimuli" causes the OSA application to be invoked in order to establish a call between the user and a customer service representative. The call being set-up can be between different entities like between two phones, a phone and an IP host, or two IP hosts.

The recommended flow is as follows: First a call object is created. Then user A's call leg is created before events are requested on it for answer and then call set-up to user A is initiated as described in the application initiated One-Party call example. On answer from user A, the call is being set up to user B. On answer from Party B the media communication between user A and user B is established..

A flow example for a Two Party call set-up from the OSA application is illustrated in the figure below:

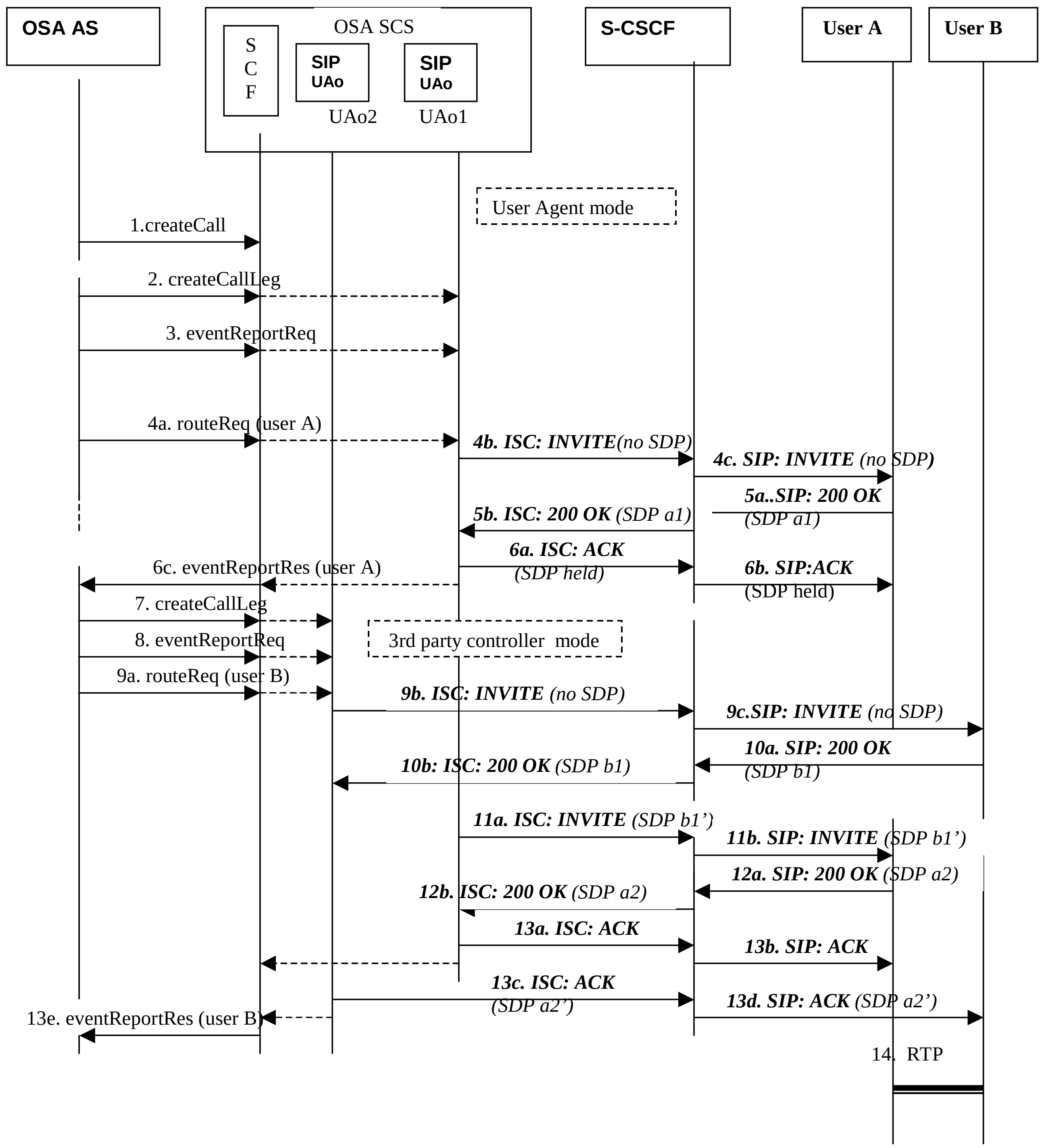

Figure B-2. Example application Initiating OSA SCS Flow for Two Party call Set-up

A description for the flow is given below:

1: through 6. Call set-up to user A. The flow is exactly the same as described in the previous example for Application initiated One-Party Call for user A.

7: This message instructs the OSA SCS (the object implementing the IpMultiPartyCall interface) to create a call leg for user B.

8: This message requests the call leg for user B to inform the application when the call leg answers the call.

9a: The created OSA terminating call leg for user B is requested to route the call/session to the specified destination for user B.

9b: The OSA SCS acting as a logical UAo2 generates an INVITE message with no SDP on the ISC interface to S-CSCF providing the destination address of user B.
The OSA SCS SIP server is now in SIP 3rd Party Controller mode (encompassing two UA Originating Endpoints, one associated with the call leg for User A and another with the call leg for user B).

9c: The S-CSCF proxies the INVITE request toward user B.

10a: User B answers the call and responds with its SDP (SIP 200 OK including SDP b1)

NOTE: It is here for simplicity assumed that the call is answered directly by user B, i.e. user B accepting the incoming call and sending a 180(Ringing) back to the UAo2 on OSA SCS is not shown.

10b: The S-CSCF proxies the SIP 200 OK including SDP b1 to the originating UAo2 in the OSA SCS via the ISC interface.

11a: The OSA SCS being the controller uses the SDP b1 in the 200 OK to generate an INVITE (re-INVITE) to the first user A. The re-INVITE is based on SDP b1, but may need to be reorganised to match up media lines with those previously applied for "SDP on hold", therefore denoted as SDP b1' when SDP is here send on the ISC interface to the S-CSCF for user A.

11b: The S-CSCF proxies the INVITE (re-INVITE with SDP b1') toward user A.

12a: User A responds in a 200 OK with its SDP (SIP 200 OK including SDP a2)
Note: SDP a2 may be different from SDP a1 reported initially from user A.

12b: The S-CSCF proxies the SIP 200 OK including SDP a2 to the originating UAo1 in the OSA SCS via the ISC interface.

13a: The OSA SCS being the controller immediately generates an ACK for user A being send on the ISC interface to the S-CSCF.

13b: The S-CSCF proxies the ACK toward user A.

13c: The SDP a2 received in 200 OK from user A is to be passed immediately to user B. It may also need reorganization to match up media lines, i.e. therefore here denoted a2'. The OSA SCS being the controller generate an ACK with SDP a2' for user B being send on the ISC interface to the S-CSCF.

13d: The S-CSCF proxies the ACK with SDP a2' toward user B.

13e: The leg object (implementing user B's IpCallLeg interface) for user B in OSA SCS passes the result of the call being answered back to the application.

14: The media communication between user A and user B has been established based on exchanged SDP information.

General Remarks:

This first part of the flow is exactly as the one described previously for a One-Party Call.

The call flow is somewhat complicated as the OSA SCS acting as controller needs to perform some SDP manipulation as the call is requested to be set-up to B. The OSA SCS needs to perform some SDP manipulations. Specifically, it must take some SDP, and generate another SDPwhich has the same media composition, but is on hold. Secondly, it may need to reorder an SDP x, so that its media lines match up with those in some other SDP y.

However, still the OSA SCS does not need to assume anything about the supported media of the terminals. There should be no problem with timers as it must be expected that a re-INVITE will be answered quickly. As we make a re-INVITE we cannot assume anything about the SDP that will be send back in the 200 OK, that is also why no SDP is used in the initiating INVITE for user B.

Once the two party call has been established, the OSA SCS operation in 3rd party controller mode is still a central point for signalling control, it now has complete control over the call. It can e.g. on request from the application disconnect one user, disconnect all users (i.e. the call), reconnect one user to another user (e.g. a follow-on call) or connect a user to another user being e.g. a media server for an announcement or conference call.

NOTE: One issue worth mentioning is the case of a follow on call where the leg for the new callee is ringing (180) or is rejected e.g. busy (e.g. 486 "Busy Here") and the application wants this information to be conveyed to the caller. Since the OSA application initiated the call set-up this information cannot be propagated by the OSA SCS toward the caller. However, one way to inform the caller could be by connection of the user (caller) to a media server for e.g. an announcement or tone sending.

Once the calls are established, both user A and user B believe they are in a single point-to-point call with some control system (assuming the OSA SCS has identified itself as the controller in the From field of the INVITE). However, they are exchanging media directly with each other, rather than with the controller, here the OSA SCS. The result is that the OSA application has set up a call between user A and user B.

B.5 Example OSA SCS control of User initiated Two-Party Call

An example of an application controlled user initiated Two-Party Call could be a Call Forwarding service. The call being set-up can be between different entities like between two phones, a phone and an IP host, or two IP hosts.

An example flow for a user initiated Two Party call set-up controlled from the OSA application is depicted in the figure below:

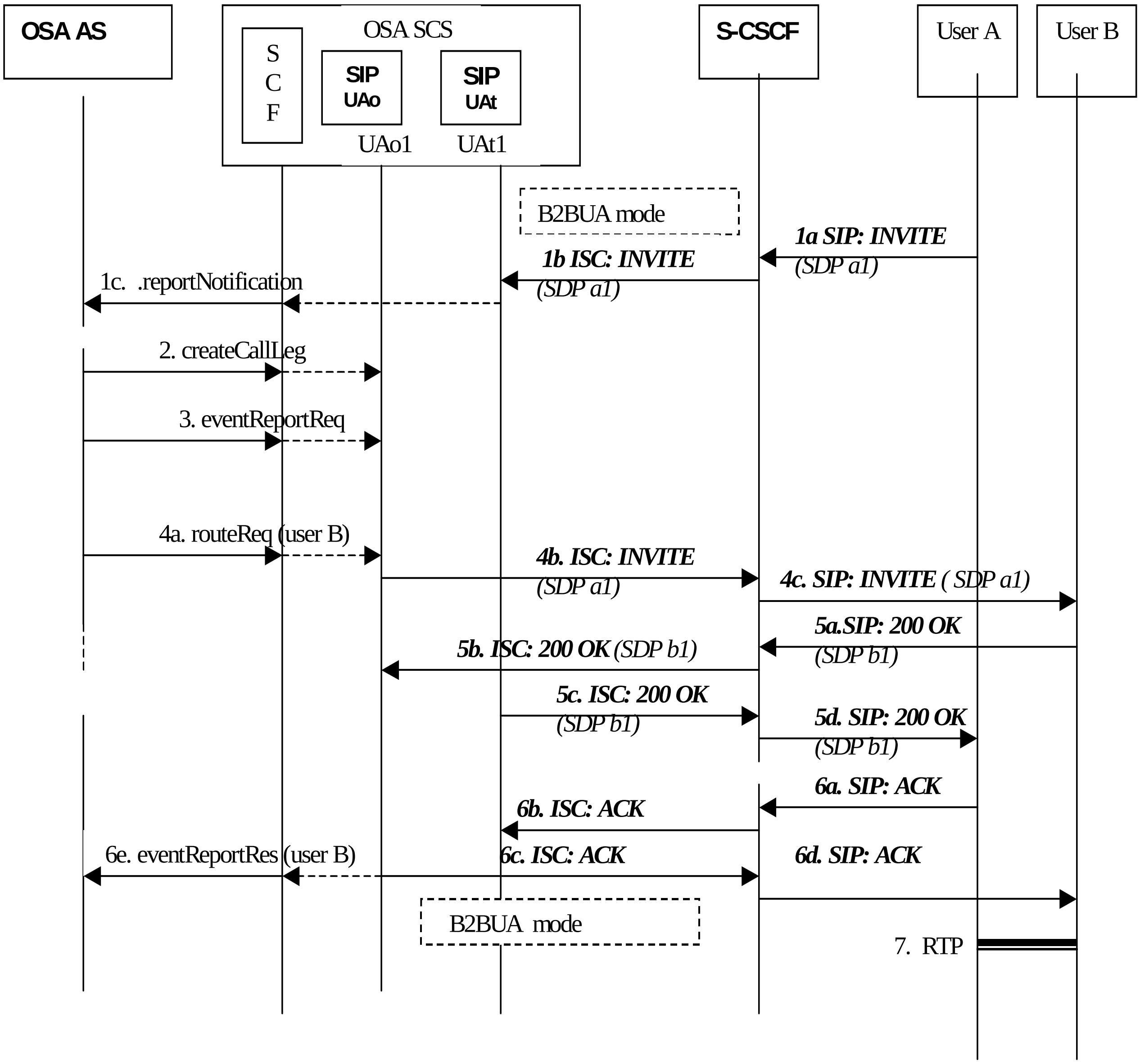

Figure B-3: Example user Initiating OSA SCS Flow for Two Party call Set-up

A description for the flow is given below:

1a: The S-CSCF receives the incoming invitation (INVITE) from user A for a dialog. As the initial filtering identifies the need to invoke an application, the S-CSCF proxies the INVITE to the OSA SCS via the ISC interface.

1b: The OSA SCS receives the incoming INVITE via the ISC interface. As the application to be invoked demands B2BUA mode of operation (i.e. to secure full call/session control), the OSA SCS is acting as a logical User Agent (UAt1) for the incoming INVITE message received from the S-CSCF. The OSA SCS creates an OSA call object (the object implementing the IpMultiPartyCall interface) and a leg object (implementing user A's

IpCallLeg interface). The leg object represents the OSA originating call leg for user A, i.e. the leg defined by the OSA MPCCS API on which the dialog invitation is received (i.e. the initial INVITE).

1c: The OSA SCS identifies the application responsible for handling the call .The application is invoked with this message to the OSA AS. The created call object and call leg object are passed to the application.

2: This message instructs the OSA SCS (e.g. the object implementing the IpMultiPartyCall interface) to create a call leg for user B.

3: This message requests the call leg for user B to inform the application when the call leg answers the call.

4a: The created terminating call leg for user B is requested to route the call/session to the specified destination for user B.

4b: The OSA SCS acting as a logical User Agent (UAo1) proxies (after some modification) the received INVITE message on the ISC interface to S-CSCF providing the destination address for user B.
The OSA SCS SIP server is now in Back-to-Back User Agent (B2BUA)mode (hereby encompassing a UA Terminating Endpoint associated with the call leg (SIP dialog) for User A and another UA Originating Endpoint associated with the call leg (SIP dialog) for user B).

4c: The S-CSCF proxies the INVITE request toward user B.

5a: User B answers the call and responds with its SDP (SIP 200 OK including SDP b1)
Note: It is here for simplicity assumed that the call is answered directly by user B, i.e. user B accepting the incoming call and sending a 180(Ringing) back to the UAo1 in OSA SCS is not shown.

5b: The S-CSCF proxies the SIP 200 OK including SDP b1 to the originating UAo1 in the OSA SCS via the ISC interface.

5c: The OSA SCS being the controller "proxies" via its terminating UAt1 the SIP 200 OK including SDP b1 on the ISC interface to the S-CSCF.

5d: The S-CSCF proxies the 200 OK (with SDP b1) toward user A.

6a: User A responds with an ACK

6b: The S-CSCF proxies the ACK to the terminating UAt1 in the OSA SCS via the ISC interface.

6c: The OSA SCS "proxies" via its originating UAo1 the ACK on the ISC interface to the S-CSCF.

6d: The S-CSCF proxies the ACK toward user B.

6e: The leg object (implementing party B's IpCallLeg interface) for user B in OSA SCS passes the result of the call being answered back to the application.

7: The media communication between user A and user B has been established based on exchanged SDP information.

General Remarks:

Once the two party call has been established, the OSA SCS as the controller is exactly in the same state as if it had initiated the call on request from the OSA application as described in a previous flow example.
The OSA SCS operation in B2BUA (or 3rd party controller) mode provides a central point for signalling control, as the application hereby is offered complete control over the call. The application can e.g. disconnect one user, disconnect all users (i.e. the call), reconnect one user to another user (e.g. a follow-on call) or connect a user to a specialised user (e.g. a user representing media server for an announcement or call conference).

B.6 Example OSA SCS control of User initiated Two-Party Call with announcement

The flow for a two –party call may also be extended so that an announcement could also be played e.g. to user A after the call with user B has been established. The announcement can be accomplished by setting up a SIP call session to a user C (e.g. being an IP host representing a media server (MRF)).
While the announcement is being played, user B's media stream is put on hold. After the announcement has been played (e.g. determined by a predefined timeout) the application may cancel the announcement and release user C (the media server represented by the MRF) and re-establish the call between user A and user B including the media communication (exchange of SDP information).

An example of an application controlled possible connection of a media server to a user on an already established Two-Party Call is depicted in the flow below:

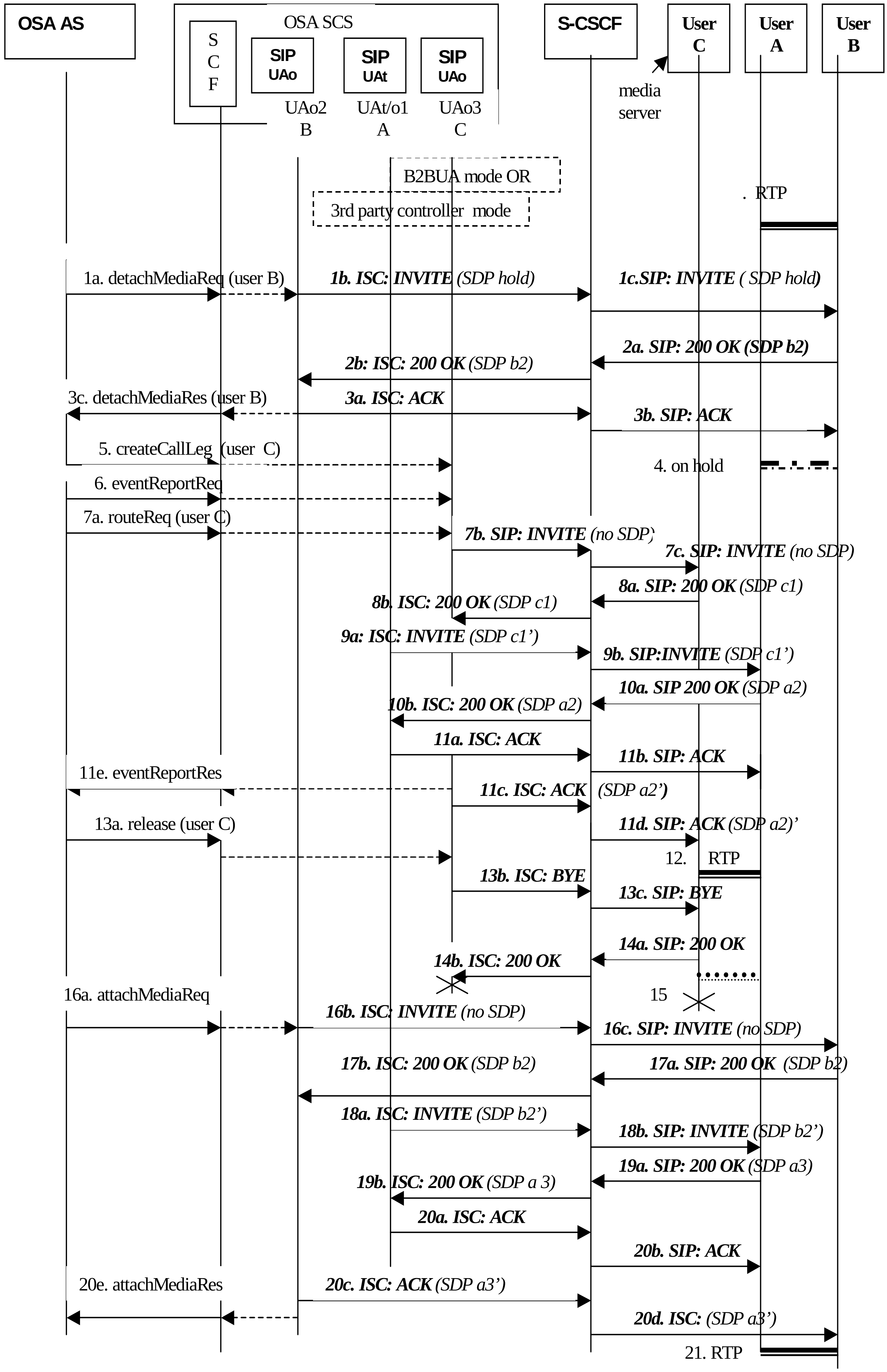

Figure B-4. Example application Initiating call to media server on a Two Party call

A description for the flow is given below:

1a: This message instructs the leg object (implementing party B's IpCallLeg interface) for user B in OSA SCS to detach the call leg from the call, i.e. prevent transmission for user B of any media streams to and from other parties in the call.

1b: The OSA SCS acting as a logical User Agent (UAo2) generates an INVITE (re-INVITE) with "SDP on hold" for user B. The re-INVITE is sent on the ISC interface to the S-CSCF.

1c: The S-CSCF proxies the INVITE (re-INVITE with SDP on hold) toward user B.

2a: User B responds in a 200 OK with its SDP (SDP b2).

NOTE: SDP b2 may be different from SDP b1 reported initially from user B during call establishment.

2b: The S-CSCF proxies the SIP 200 OK (including SDP b2) to the originating UAo2 in the OSA SCS via the ISC interface.

3a: The OSA SCS being the controller immediately generates from UAo2 an ACK for user B being send on the ISC interface to the S-CSCF.

3b: The S-CSCF proxies the ACK toward user B.

3c: The leg object (implementing party B's IpCallLeg interface) for user B in OSA SCS passes the result of the call leg being detached back to the application.

4: The media communication for user B is on hold.

5: This message instructs the OSA SCS (e.g. the object implementing the IpMultiPartyCall interface) to create a call leg for user C.

6: This message requests the call leg for user C to inform the application when the call leg answers the call.

7a: The created OSA terminating call leg for user C is requested to route the call/session to the specified destination for user C.

7b: The OSA SCS acting as a logical UAo3creates an INVITE message (with no SDP) on the ISC interface to S-CSCF providing the destination address of user C.
The OSA SCS SIP server is now in SIP 3rd Party Controller mode (encompassing three UAs).

7c: The S-CSCF proxies the INVITE request toward user C.

8a: User C answers the call and responds with its SDP (SIP 200 OK including SDP c1).

NOTE: It is here for simplicity assumed that the call is answered directly by user C, i.e. user C accepting the incoming call and sending a 180(Ringing) back to the UAo3 in OSA SCS is not shown.

8b: The S-CSCF proxies the SIP 200 OK including SDP c1 to the originating UAo3 in the OSA SCS via the ISC interface.

9a: The OSA SCS being the controller uses the SDP c1 in the 200 OK to generate an INVITE (re-INVITE) to user A. The re-INVITE is based on SDP c1, but may need to be reorganised to match up media lines with those previously applied, therefore denoted as SDP c1' when SDP is send on the ISC interface to the S-CSCF for user A.

9b: The S-CSCF proxies the INVITE (re-INVITE with SDP c1') toward user A.

10a: User A responds in a 200 OK with its SDP (SIP 200 OK including SDP a2).

NOTE: SDP a2 may be different from SDP a1 reported initially from user A during call establishment.

10b: The S-CSCF proxies the SIP 200 OK(including SDP a2) to the originating/terminating UAo1/UAt1 in the OSA SCS via the ISC interface.

11a: The OSA SCS being the controller immediately generates an ACK for user A being send on the ISC interface to the S-CSCF.

11b: The S-CSCF proxies the ACK toward user A.

11c: The SDP a2 received in 200 OK from user A is to be passed immediately to user C. It may also need reorganization to match up m lines, i.e. therefore here denoted a2'. The OSA SCS being the controller generate an ACK with SDP a2' for user C being send on the ISC interface to the S-CSCF (response to 200 OK in 8b).

11d: The S-CSCF proxies the ACK with SDP a2' toward user C.

11e: The leg object (implementing party C's IpCallLeg interface) for user C in OSA SCS passes the result of the call being answered back to the application.

12: The media communication between user A and user C has been established based on exchanged SDP information.

13a: This message instructs the leg object (implementing party C's IpCallLeg interface) for user C in OSA SCS to release the call leg from the call.

13b: The OSA SCS acting as a logical UAo3 issues the BYE message on the ISC interface to S-CSCF for the release of user C.

14a: User C responds in a 200 OK.

14b: The S-CSCF proxies the SIP 200 OK to the originating UAo3 in the OSA SCS via the ISC interface. The UAo3 and the call leg object for C is terminated (destroyed).

15: The media communication between user A and user C is terminated.

16a: This message instructs the leg object (implementing party B's IpCallLeg interface) for user B in OSA SCS to attach the call leg for user B to the call to enable any media streams to and from other parties in the call.

16b: The OSA SCS acting as a logical User Agent (UAo2) generates an INVITE (re-INVITE with no SDP) for user B. The re-INVITE is sent on the ISC interface to the S-CSCF.

16c: The S-CSCF proxies the INVITE (re-INVITE with no SDP) toward user B.

17a: User B responds in a 200 OK with its SDP (SDP b2).

NOTE: SDP b2 may be different from SDP b1 reported initially from user B during call establishment.

17b: The S-CSCF proxies the SIP 200 OK (including SDP b2) to the originating UAo2 in the OSA SCS via the ISC interface.

18a: The OSA SCS being the controller uses the SDP b2 in the 200 OK from user B to generate an INVITE (re-INVITE) from UAo1/UAt1 to user A. The re-INVITE is based on SDP b2, but may need to be reorganised to match up media lines with those previously applied , therefore denoted as SDP b2' when SDP is send on the ISC interface to the S-CSCF for user A.

18b: The S-CSCF proxies the re-INVITE toward user A.

19a: User A responds in a 200 OK with its SDP (SDP a3).

NOTE: SDP a3 may be different from SDP a1 reported initially from user A during call establishment.

19b: The S-CSCF proxies the SIP 200 OK (including SDP a3) to the UAo1/UAt1 in the OSA SCS via the ISC interface.

20a: The OSA SCS being the controller immediately generates an ACK for user A being send on the ISC interface to the S-CSCF.

20b: The S-CSCF proxies the ACK toward user A.

20c: The SDP a3 received in 200 OK from user A is to be passed immediately to user B. It may also need reorganization to match up m lines, i.e. therefore here denoted a3'. The OSA SCS being the controller generate an ACK with SDP a3' for user B being send from UAo2 on the ISC interface to the S-CSCF (response to 200 OK in 17b).

20d: The S-CSCF proxies the ACK with SDP a3' toward user B.

20e: The leg object (implementing party B's IpCallLeg interface) for user B in OSA SCS passes the result of the call leg being attached back to the application.

21: The media communication between user A and user B has been re-established based on exchanged SDP information.

General Remarks:

The flow 5- 12 for call set-up to C party is exactly the same as for the call set-up to B-party.

Flow 1-4 and 16-21: Different implementation options may apply for attach/detach media; in the flow example above it is anticipated that the OSA SCS would not re-use (store) any SDP information previously received from the users, but always fetch it when needed, i.e. for detachMediaReq / attachMediaReq always retrieve the actual SDP information from the user (with SDP in 200 OK in response to re-INVITE).
Another option could also be to preference re-INVITE with no SDP and so for attach media provide the SDP within the ACK (instead of including the SDP in the re-INVITE itself as shown in the flow).

B.7 Example OSA SCS Application initiated Multi-Party Call

The capability to control multiple call legs is supported by the MPCCS. The OSA SCS when acting as 3rd. party controller can create and control multiple call-legs (i.e. more than two parties involved in a call).

The 2-party call may as a variation be extended to include 3 parties (or more). After a two party call is established, the application can create a new leg and request to route it to a new destination address in order to establish a 3 party call.

The event that causes this to happen could for example be the report of answer event from B-party or controlled by the A-party by entering a service code (mid-call event) or some other stimuli.

Furthermore conference call may be established by connection each user to a "specialized" user, i.e. a conference device represented by a MRF entity, but addressed like any other user via SIP. Hereby a conference call could be established as a set of two party calls where each call is termination at the same "user", i.e. the user (MRF) constituting the conference device in the network.

NOTE: Recommended call flows for such a 3-party call scenarios etc. should be provided in this section to especially describe the handling of SDP in case of multiple parties in a call session. This is for further study.

Annex C: OSA call forwarding presentation

C.1 Introduction

The application can request a call forwarding causing a SIP session being forwarded to a new destination. The applied methods for this (createAndRouteCallLegReq and routeReq) specifies that in case the application wants the call to be presented in the network as a redirection (call forwarding) it should include the Original Destination Address. The same should apply for the presence of field REDIRECTING_ADDRESS in AppInfo.

The question raised is how to present this to callee and caller, i.e. make the call visible in the network as a redirected or forwarded call.

When the application instructs a call redirection containing beside the targetAddress (SIP URL) parameter also the Original Destination Address (field in TpCallAppInfo) and / or Redirecting Address the call is to be presented in the network as being a redirection, e.g. in case of any call forwarding service.

C.2 Call Forwarding presentation in OSA: mapping to SIP

The following mappings to SIP applies:

Toward callee:

Call redirection information is to be given to the callee (forwarded-to- party) so that this callee may respond to the caller appropriately. In these situations, the party receiving a redirected call needs an answer to the questions:

Q1: From whom was the request diverted?

Q2: Why was the request diverted?

The SIP Diversion header is used to answer these questions for the party receiving the diverted call.

First the reply to Q1 is given:

Original Destination Address:
In response to createAndRouteCallLegReq and routeReq if the **Original Destination Address** is present there shall be a map of the redirecting address to the Diversion header being added to the SIP INVITE.
As the INVITE request may contain information about the first and subsequent redirections
the Original Destination Address, when present, should be used to set the bottom-most Diversion header to present the original called address (if not already inserted here).

Redirecting address:
How to map the presence of field REDIRECTING_ADDRESS in appInfo in response to createAndRouteCallLegReq and routeReq. This field contains the address of the user from which the call is redirected /diverted

Here the top-most Diversion header is to be used to set the Redirecting address.

reply to Q2:

Information regarding why the call request was diverted is given by filling in the "reason" tag into the Diversion header (by the OSA SCS). Here a default value "unknown" is recommended as "diversion-reason".

NOTE 1: Currently there is no MPCCS API support allowing the application to indicate "diversion-reason". The diversion-reason should be used to set the Redirecting Reason corresponding to the associated redirecting addressinserted into the SIP Diversion header field.

NOTE 2: A Diversion header is added when features such as call forwarding change the Request-URI.
The proposal herein is in alignment with how redirection numbers are mapped between ISUP and SIP.

Toward caller:
To make the call visible as a forwarded call in the network the provisional response 181 "Call Is Being Forwarded "should be sent upstream by the SIP proxy (e.g. the OSA SCS gateway). This response is to indicate to the caller that the call is being forwarded to a different (set of) destination(s).

targetAddress :
The targetAddress received in createAndRouteCallLegReq and routeReq should be included in the 181 provisional response as to enable the presentation of the "forwarded to" address to the caller, i.e. the current destination address. redirected address.

NOTE 3: If the call is a call redirection, i.e. the appInfo should include at least one of the fields: ORIGINAL_DESTINATION_ADDRESS and/or REDIRECTING_ADDRESS as to identify the routing request to be a request for a call redirection. In this case the OSA SCS should store the targetAddress as to enable the application to use getCurrentDestinationAddress to read the address where the call was directed to. This address is also to be sent upstream in a 181 provisional response to enable previous invoked applications as well as the caller to be notified.

NOTE 4: A previous invoked application (further upstream) should then be notified of the call being forwarded if it has subscribed to the event CALL_EVENT_REDIRECTED including the redirected address (forwardAddress).

NOTE 5: The redirected address (i.e. the current address of the termination point) is to be stored in the OSA SCS so that the application can request this information anytime with the getCurrentDestinationAddress.

Annex D (informative): Description of Multiparty Call Control ISC Mapping for 3GPP2 cdma2000 networks

This annex is intended to define the OSA API Stage 3 interface definitions and it provides the complete OSA specifications. It is an extension of OSA API specifications capabilities to enable operation in cdma2000 systems environment. They are in alignment with 3GPP2 Stage 1 requirements and Stage 2 architecture defined in:

[1] 3GPP2 P.S0001-B: "Wireless IP Network Standard", Version 1.0, September 2000.

[2] 3GPP2 S.R0037-0: "IP Network Architecture Model for cdma2000 Spread Spectrum Systems", Version 2.0, May 14, 2002.

[3] 3GPP2 X.S0013: "All-IP Core Network Multimedia Domain", December 2003.

These requirements are expressed as additions to and/or exclusions from the 3GPP Release 6 specification.
The information given here is to be used by developers in 3GPP2 cdma2000 network architecture to interpret the 3GPP OSA specifications.

D.1 General Exceptions

- The term UMTS is not applicable for the cdma2000 family of standards. Nevertheless these terms are used (3GPP TR 21.905) mostly in the broader sense of "3G Wireless System". If not stated otherwise there are no additions or exclusions required.
- CAMEL and CAP mappings are not applicable for cdma2000 systems.

D.2 Specific Exceptions

Clause 1: Scope

- There are no additions or exclusions.

Clause 2: References

- There are no additions or exclusions.

Clause 3: Definitions and abbreviations

- There are no additions or exclusions.

Clause 4: Mapping OSA Call and Call Leg to SIP

- There are no additions or exclusions.

Clause 5: Multi Party Call Control Flows

- There are no additions or exclusions.

Clause 6:Detailed parameter mappings

- There are no additions or exclusions.

Annex A: Introduction to API Mapping for OSA MPCCS

- There are no additions or exclusions.

Annex B: SDP in SIP at application controlled calls for OSA MPCCS API

- There are no additions or exclusions.

Annex C: OSA call forwarding presentation

- There are no additions or exclusions.

Annex E: Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
April 2002	--	--	--	--	Draft v100 submitted to TSG CN email list for Information	--	1.0.0
Jun 2002	CN_16	NP-020197	--	--	Draft v200 submitted to TSG CN#16 for Approval	2.0.0	5.0.0
Dec 2003	CN_22	NP-030553	001	--	Add OSA API support for 3GPP2 networks in ISC Mapping	5.0.0	6.0.0
Mar 2004	--	--	--	--	Updated references to IETF ([14], [16])	6.0.0	6.0.1
Apr 2004	--	--	--	--	Updated reference [16]. Reason: RFC# allocated by IETF (Musa).	6.0.1	6.0.2
Jun 2004	--	--	--	--	Updated reference [16]. Reason: RFC agreed by IETF (John-Luc)	6.0.2	6.0.3
Dec 2004	CN_26	--	--	--	Updated Introduction (changed SPAN12 to TISPAN, added Part 15 to OSA API family), converted in References TS 22.121 to 23.198 (the new OSA Stage 2 TS), modified release 5 unsolved issue to a release independent formulation.	6.0.3	6.0.4
Mar 2007	CT_35	--	--	--	Automatic upgrade to R7 (no CR needed)	6.0.0	7.0.0

History

Document history		
V7.0.0	March 2007	Publication