3GPP TSG CN Plenary Meeting #13 **NP-010467**
Beijing, China, 19th –21st September 2001

Source: **CN5 (OSA)**

Title: **CRs 29.198-04 Rel-4**

Agenda item: **8.5**

Document for: **Approval**

Doc-1st-Level	Doc-2nd-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Version-New	Meeting-2nd-Level	Workitem
NP-010467	N5-010677	29.198-04	001		Rel-4	Changing references to JAIN	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010532	29.198-04	002		Rel-4	Correction of text descriptions for methods enableCallNotification and createNotification	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010533	29.198-04	003		Rel-4	Specify the behaviour when a call leg times out	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010542	29.198-04	004		Rel-4	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010543	29.198-04	005		Rel-4	Missing TpCallAppInfoSet description in OSA R4	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010612	29.198-04	006		Rel-4	Redirecting a call leg vs. creating a call leg clarification in OSA R4	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010614	29.198-04	007		Rel-4	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010631	29.198-04	008		Rel-4	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010635	29.198-04	009		Rel-4	Corrections to SetChargePlan()	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010640	29.198-04	010		Rel-4	Remove distinction between final- and intermediate-report	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010648	29.198-04	011		Rel-4	Inclusion of TpMediaType	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010660	29.198-04	012		Rel-4	Corrections to GCC STD	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010693	29.198-04	013		Rel-4	Introduction of sequence diagrams for MPCC services	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010707	29.198-04	014		Rel-4	The use of the REDIRECT event needs to be illustrated	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010709	29.198-04	015		Rel-4	Corrections to SetCallChargePlan()	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010710	29.198-04	016		Rel-4	Add one additional error indication	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010633	29.198-04	017		Rel-4	Corrections to Call Control – GCCS Exception handling	F	4.0.0	4.1.0	N5-12	OSA1
NP-010467	N5-010704	29.198-04	018		Rel-4	Corrections to Call Control – Errors in Exceptions	F	4.0.0	4.1.0	N5-12	OSA1

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010677**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **001** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Changing references to JAIN
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	Incorrect references to JAIN.
Summary of change:	⌘	Correct references to the JAIN.
Consequences if not approved:	⌘	Potential legal ramifications

Clauses affected:	⌘	1		
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	All other parts of TS 29.198 Rel-4
Other comments:	⌘			

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1 Scope

The present document is Part 4 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Call Control Service Capability Feature (SCF) aspects of the interface. All aspects of the Call Control SCF are defined here, these being:

- Sequence Diagrams
- Class Diagrams
- Interface specification plus detailed method descriptions
- State Transition diagrams
- Data definitions
- IDL Description of the interfaces

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI SPAN 12 and the Parlay Consortium, in co-operation with a number of JAIN™ Community member companies~~the JAIN consortium~~.

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **002** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction of text descriptions for methods enableCallNotification and createNotification
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	When a notification is requested through enableCallNotification in GCC (or createNotification in MPCC), the descriptions for the methods currently state that overlap checking on the notification's criteria will take place to determine if the request will be allowed to proceed. The descriptions do not however specify that if the 'monitor mode' in the criteria is set to 'notify', then the overlap checking procedure (on the rest of the criteria) will not be necessary as any application can freely request to be notified. The descriptions to the method also do not specify that there can only be one application placing an interrupt request at a time.
Summary of change:	⌘	Lucent proposes to edit the descriptions of the methods enableCallNotification (in GCC) and createNotification (in MPCC) to include some text specifying that if a request for notification is made with 'notify' monitor mode, then overlap checking will not be needed, and also specifying that only one application can place an interrupt request if the criteria overlaps.
Consequences if not approved:	⌘	There will be no explicit statement that only one INTERRUPT request can be placed at a time when the criteria overlaps, resulting in ambiguous behaviour. Failure to adopt this CR would result in divergence between the 3GPP R4 specification and the ETSI/Parlay specifications.

Clauses affected:	⌘	6.3.1, 7.3.1
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	
Other comments:	⌘	

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

Problem

When a notification is requested through enableCallNotification in GCC (or createNotification in MPCC), descriptions to the methods currently state that if there is another request comes along, an overlap checking on the notification's criteria will take place to determine if the request will be allowed to proceed. The descriptions do not however specify that if the 'monitor mode' in the criteria is set to 'notify', then overlap checking procedure (on the rest of the criteria) will not be necessary as any application can freely request to be notified. The descriptions to the method also do not specify that there can only be one application placing an interrupt request at a time.

Furthermore, the description to the method createNotification in Multi-Party Call Control interfaces makes use of P_GCCS_INVALID_CRITERIA which is an exception code meant to be used only in Generic Call Control interfaces.

Proposal

Lucent proposes to edit the descriptions of the methods enableCallNotification (in GCC) and createNotification (in MPCC) to include some text specifying that if a request for notification is made with 'notify' monitor mode, then overlap checking will not be needed, also specifying that only one application can place an interrupt request if the criteria overlaps.

The proposal is also to correct P_GCCS_INVALID_CRITERIA in the description to method createNotification in MPCC interfaces as it should be P_INVALID_CRITERIA.

Result changes

6.3.1 Interface Class IpCallControlManager

Method

`enableCallNotification()`

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notification of calls happening in the network. When such an event happens, the application will be informed by callEventNotify(). In case the application is interested in other events during the context of a particular call session it has to use the routeReq() method on the call object. The application will get access to the call object when it receives the callEventNotify(). (Note that the enableCallNotification() is not applicable if the call is setup by the application).

The enableCallNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P_GCCS_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same CallNotificationType is used.

If a notification is requested by an application with the monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not allow control on a call to be passed over. Only one application can place an interrupt request if the criteria overlaps.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most

recent is used. In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallBack().

Parameters

`appCallControlManager : in IpAppCallControlManagerRef`

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

`eventCriteria : in TpCallEventCriteria`

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

`assignmentID : out TpAssignmentIDRef`

Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Raises

TpGCCSException,TpGeneralException

7.3.1 Interface Class IpMultiPartyCallControlManager

Method

`createNotification()`

This method is used to enable call notifications so that events can be sent to the application. This is 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(). In case the application is interested in other events during the context of a particular call session it has to use the createAndRouteCallLegReq() method on the call object or the eventReportReq() method on the call leg object. The application will get access to the call object when it receives th~~y~~e reportNotification(). (Note that createNotification() is not applicable if the call is setup by the application).

The createNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with ~~P_GCCS_INVALID_CRITERIA.~~ P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same NotificationCallType is used.

If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not allow control on a call to be passed over. Only one application can place an interrupt request if the criteria overlaps.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Parameters

`appCallControlManager : in IpAppMultiPartyCallControlManagerRef`

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

notificationRequest : in TpCallNotificationRequest

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

assignmentID : out TpAssignmentIDRef

Specifies the ID assigned by the ~~generic~~ multi party call control manager interface for this newly-enabled event notification.

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010533**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **003** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

Title: ⌘	Specify the behaviour when a call leg times out	
Source: ⌘	CN5	
Work item code: ⌘	OSA1	***Date:*** ⌘ 30/08/2001
Category: ⌘	**F**	***Release:*** ⌘ REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change: ⌘	The behaviour of the call when the activity timer expires has been specified. However, the behaviour of the call leg when its activity timer expires has not yet been specified.
Summary of change: ⌘	It is proposed that the callLegEnded/connectionEnded method on IpAppCallLeg is invoked when the activity timer expires for a call leg, giving a reason indicating the expiry of a timer. Also, one of the parameters to this method is a release cause of type TpCallReleaseCause. This is not an appropriate type for the call leg as the call itself may still be up. Lucent would like to propose that the TpCallReleaseCause is renamed to TpReleaseCause for MPCC. All references to TpCallReleaseCause will be replaced.
Consequences if not approved: ⌘	The entire call will be ended if the activity timer on one of the call legs expires. Failure to adopt this CR would result in divergence between the 3GPP R4 specification and the ETSI/Parlay specifications.

Clauses affected: ⌘	7.3.3, 7.3.5, 7.3.6, 7.4.2, 7.6.2, 10	
Other specs affected: ⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘
Other comments: ⌘		

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at:
http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

There is a remaining part of contribution N5-010364 (reworked as N5-010461 in San Diego) which has not yet been implemented due to the definition of the call leg being incomplete:

In the text that describes the activity timers for the IpMultiPartyCall, it should be mentioned that the action upon expiry of the activity timer is to invoke callEnded() on the IpAppMultiPartyCall with a release cause of P_TIMER_EXPIRY. It should also state that in the case when no IpAppMultiPartyCall is available on which to invoke callEnded(), callAborted() shall be invoked on the IpAppMultiPartyCallControlManager as this is an abnormal termination.

Resulting Changes

7.6.2 Multi-Party Call Control Data Definitions

TpAdditionalCallEventCriteria

Defines the Tagged Choice of Data Elements that specify specific criteria.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpInt32	MinAddressLength
P_CALL_EVENT_ADDRESS_ANALYSED	NULL	Undefined
P_CALL_EVENT_PROGRESS	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_RELEASE	Tp~~Call~~ReleaseCauseSet	ReleaseCauseSet
P_CALL_EVENT_REDIRECTED	NULL	Undefined
P_CALL_EVENT_SERVICE_CODE	TpCallServiceCode	ServiceCode

Tp~~Call~~ReleaseCauseSet

Defines a Numbered Set of Data Elements of Tp~~Call~~ReleaseCause.

TpCallAdditionalEventInfo

Defines the Tagged Choice of Data Elements that specify additional call event information for certain types of events.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpAddress	CollectedAddress
P_CALL_EVENT_ADDRESS_ANALYSED	TpAddress	CalledAddress
P_CALL_EVENT_PROGRESS	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_RELEASE	Tp~~Call~~ReleaseCause	ReleaseCause
P_CALL_EVENT_REDIRECTED	TpAddress	ForwardAddress
P_CALL_EVENT_SERVICE_CODE	TpCallServiceCode	ServiceCode

TpCallLegInfoReport

Defines the Sequence of Data Elements that specify the call leg information requested.

Sequence Element Name	Sequence Element Type	description
CallLegInfoType	TpCallLegInfoType	The type of the call leg.
CallLegStartTime	TpDateAndTime	The time and date when the call leg was started (i.e., the leg was routed).
CallLegConnectedToResourceTime	TpDateAndTime	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	The date and time 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. Either this element is valid or the CallConnectedToResourceTime is valid, depending on whether the report is sent as a result of user interaction.
CallLegEndTime	TpDateAndTime	The date and time when the call leg was released.
ConnectedAddress	TpAddress	The address of the party associated with the leg. If during the call the connected address was received from the party 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	Tp~~Call~~ReleaseCause	The cause of the termination. May be present with P_CALL_LEG_INFO_RELEASE_CAUSE was specified.
CallAppInfo	TpCallAppInfoSet	Additional information for the leg. May be present with P_CALL_LEG_INFO_APPINFO was specified.

Tp~~Call~~ReleaseCause

Defines the reason for ~~which~~ a ~~call is~~ released~~d~~.

Name	Value	Description
P_UNDEFINED	0	The reason of release isn't known, because no info was received from the network.
P_USER_NOT_AVAILBLE	1	The user isn't available in the network. This means that the number isn't allocated or that the user isn't registered.
P_BUSY	2	The user is busy.
P_NO_ANSWER	3	No answer was received
P_NOT_REACHABLE	4	The user terminal isn't reachable
P_ROUTING_FAILURE	5	A routing failure occurred. For example an invalid address was received
P_PREMATURE_DISCONNECT	6	The user disconnected the call/call leg during the setup phase.
P_DISCONNECTED	7	~~Call~~ A disconnect was received ~~by the end user~~.
P_CALL_RESTRICTED	8	The call was subject of restrictions.
P_UNAVAILABLE_RESOURCE	9	The request could not be carried out as n~~N~~o resources w~~h~~ere available ~~to establisch the call~~.
P_GENERAL_FAILURE	10	A general network failure occurred.
P_TIMER_EXPIRY	11	The call/call leg was released because an activity timer expired.

10 Common Call Control Data Types

TpCallEndedReport

Defines the Sequence of Data Elements that specify the reason for the call ending.

Sequence Element Name	Sequence Element Type	
CallLegSessionID	TpSessionID	The leg that initiated the release of the call. If the call release was not initiated by the leg, then this value is set to –1.
Cause	Tp~~Call~~ReleaseCause	The cause of the call ending.

TpCallInfoReport

Defines the Sequence of Data Elements that specify the call information requested. Information that was not requested is invalid.

Sequence Element Name	Sequence Element Type	Description
CallInfoType	TpCallInfoType	The type of call report.
CallInitiationStartTime	TpDateAndTime	The time and date when the call, or follow-on call, was started.
CallConnectedToResourceTime	TpDateAndTime	The date and time when the call was connected to the resource. This data element is only valid when information on user interaction is reported.
CallConnectedToDestinationTime	TpDateAndTime	The date and time when the call 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. This data element is invalid when information on user interaction is reported with an intermediate report.
CallEndTime	TpDateAndTime	The date and time when the call or follow-on call or user interaction was terminated.
Cause	Tp~~Call~~ReleaseCause	The cause of the termination.

A callInfoReport will be generated at the end of user interaction and at the end of the connection with the associated

address. This means that either the destination related information is present or the resource related information, but not both.

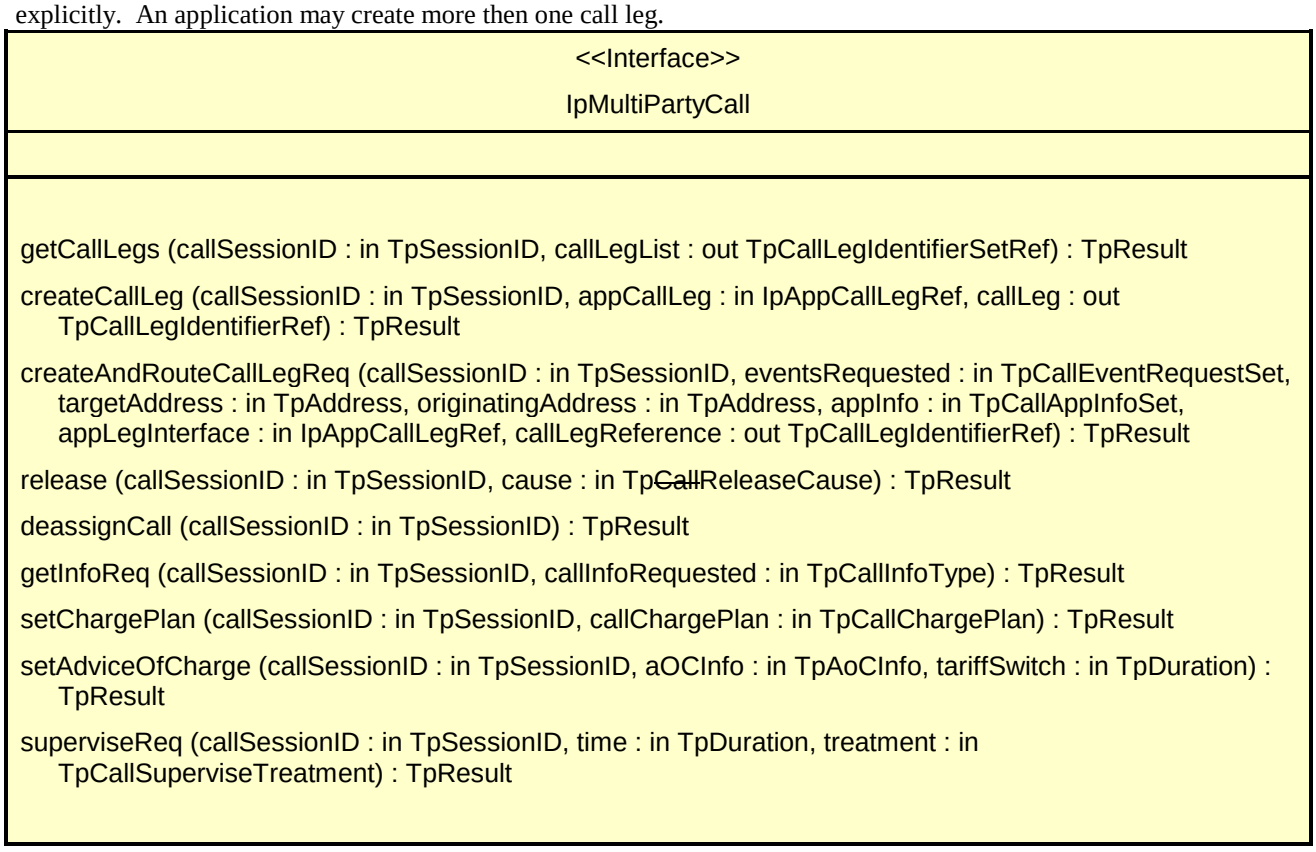
TpCallTreatment

Defines the Sequence of Data Elements that specify the the treatment for calls that will be handled only by the network (for example, call which are not admitted by the call load control mechanism).

Sequence Element Name	Sequence Element Type
ReleaseCause	TpCallReleaseCause
AdditionalTreatmentInfo	TpCallAdditionalTreatmentInfo

7.3.3 Interface Class IpMultiPartyCall

Inherits from: IpService
The Multi-Party Call provides the possibility to control the call routing, to request information from the call, control the charging of the call, to release the call and to supervise the call. It also gives the possibility to manage call legs explicitly. An application may create more then one call leg.



Method

release()

This method requests the release of the call object and associated objects. The call will also be terminated in the network. If the application requested reports to be sent at the end of the call (e.g., by means of getInfoReq) these reports will still be sent to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

cause : in Tp~~Call~~ReleaseCause

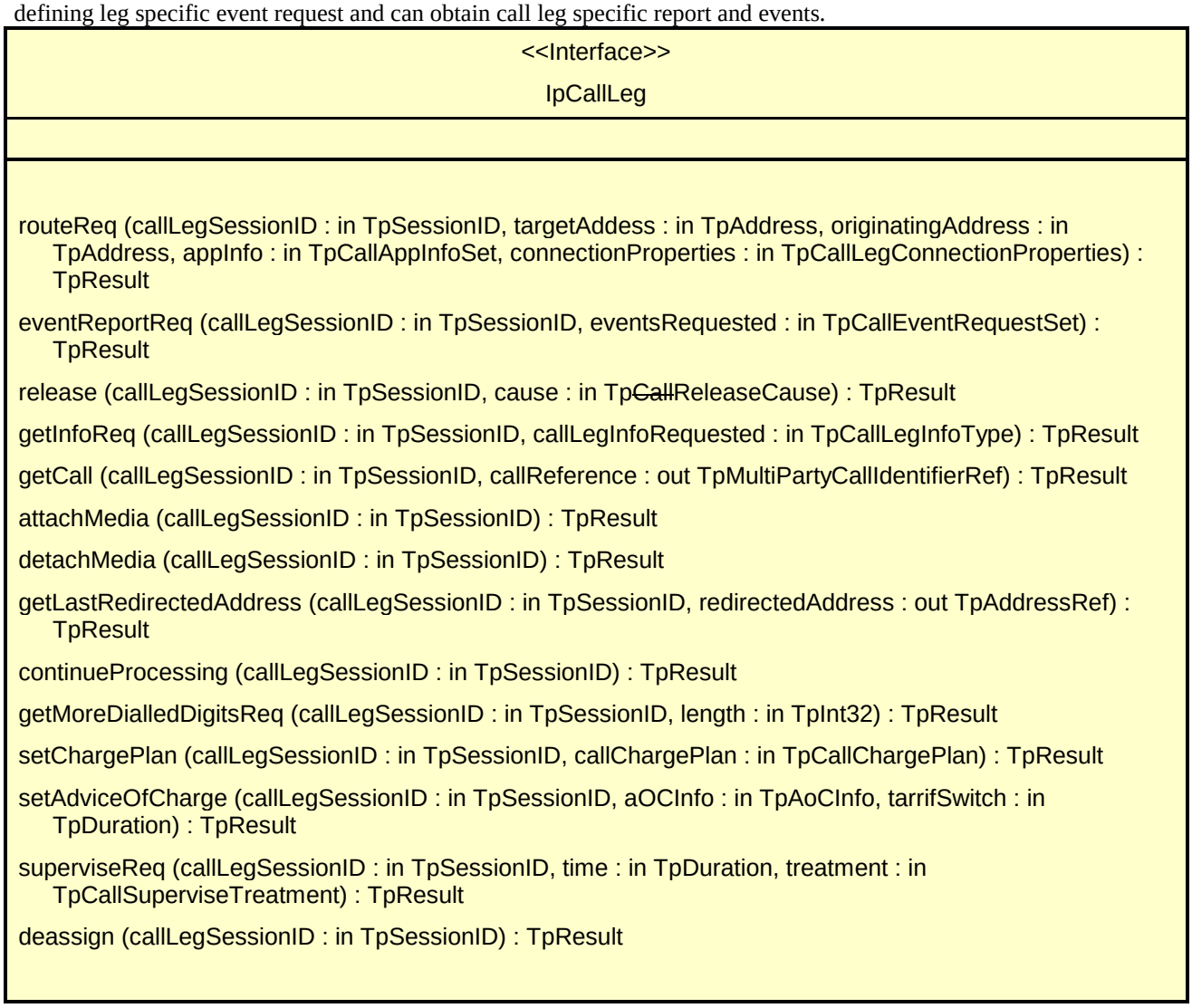
Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

7.3.5 Interface Class IpCallLeg

Inherits from: The call leg interface represents the logical call leg associating a call with an address. The call leg tracks its own states and allows charging summaries to be accessed. The leg represents the signalling relationship between the call and an address. An application that uses the IpCallLeg interface to set up connections has more control, e.g. by defining leg specific event request and can obtain call leg specific report and events.



Method

release()

This method requests the release of the call leg. If successful, the associated address (party) will be released from the call, and the call leg deleted. Note that in some cases releasing the party may lead to release of the complete call in the network. The application will be informed of this with callEnded().

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in Tp~~Call~~ReleaseCause

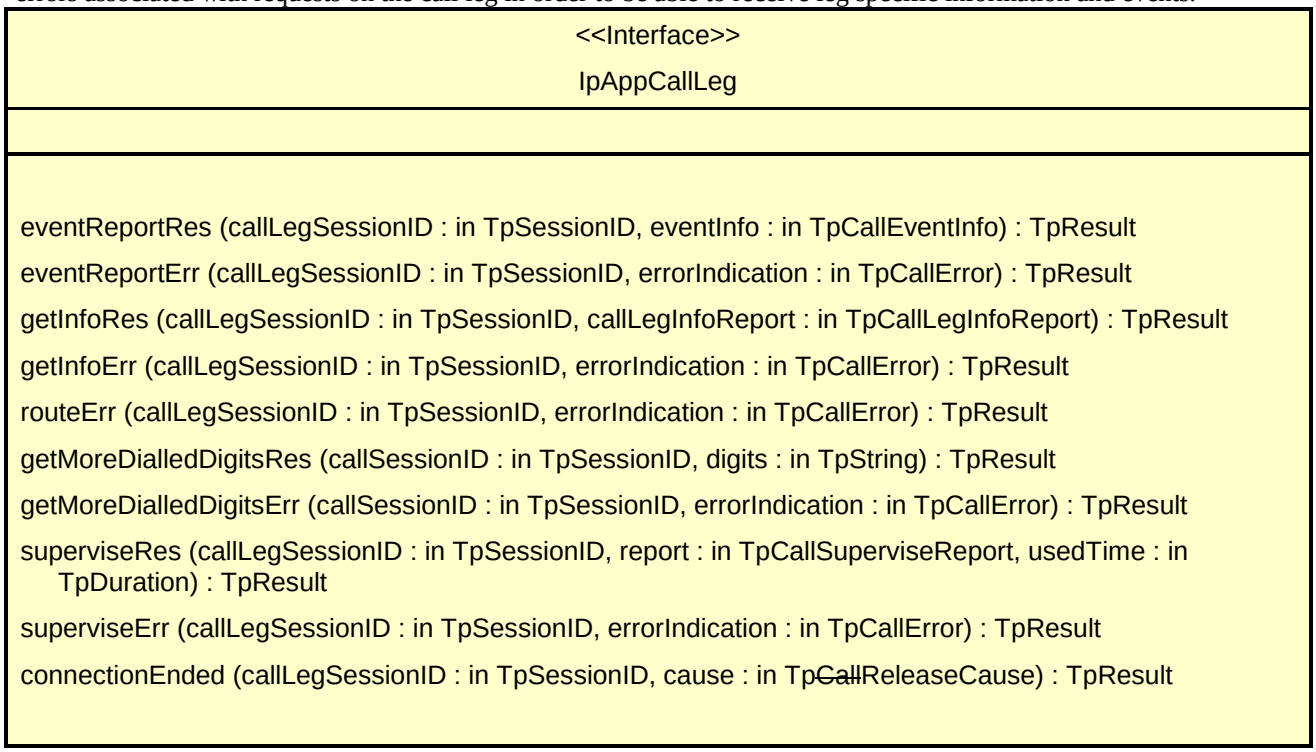
Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

7.3.6 Interface Class IpAppCallLeg

Inherits from: IpInterface
IpService
The application call leg interface is implemented by the client application developer and is used to handle responses and errors associated with requests on the call leg in order to be able to receive leg specific information and events.



Method

eventReportRes()

This asynchronous method reports that an event has occurred that was requested to be reported (for example, a mid-call event, the party has requested to disconnect, etc.).
Depending on the type of event received, outstanding requests for events are discarded. The exact details of these so-called disarming rules are captured in the data definition of the event type.
If this method is invoked for a report with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call leg. If the APL does nothing with the call leg ~~(including its associated legs)~~ within a specified time period (the duration which forms a part of the service level agreement), then the ~~call~~ connection in the network shall be released and callLegEnded() shall be invoked, giving a release cause of P_TIMER_EXPIRY.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg on which the event was detected.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

Method

connectionEnded()

This method indicates to the application that the connection has terminated in the network. However, the application may still receive some results (e.g., getInfoRes) related to the call leg. The application is expected to deassign the call leg object after having received the connectionEnded.
Note that the event that caused the connection to end might also be received separately if the application was monitoring for it.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in Tp~~Call~~ReleaseCause

Specifies the reason the connection is terminated.

7.4.2 State Transition Diagrams for IpMultiPartyCall

The state transition diagram shows the application view on the MultiParty Call object.

When an IpMultiPartyCall is created using createCall, or when an IpMultiPartyCall is given to the application for a notification with a monitor mode of P_MONITOR_MODE_INTERRUPT, an activity timer is started. The activity timer is stopped when the application invokes a method on the IpMultiPartyCall. The action upon expiry of this activity timer is to invoke callEnded() on the IpAppMultiPartyCall with a release cause of P_TIMER_EXPIRY. In the case when no IpAppMultiPartyCall is available on which to invoke callEnded(), callAborted() shall be invoked on the IpAppMultiPartyCallControlManager as this is an abnormal termination.

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010542**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **004** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	A note added to the Call State Transition Diagram invalidated certain transitions.
Summary of change:	⌘	A note was added in the Call State Transition Diagram that specifies behaviour upon TIMER EXPIRY. Through removing the FAULTY state and the callFaultDetected method and introducing callLegEnded on all transitions to the 'sink', this note is reflected in the Call State Transition Diagram. As a result the TpCallFault type is moved from the common types to the GCCS types section.
Consequences if not approved:	⌘	If these modifications are not accepted, the specification will contain ambiguities and will lead to interworking problems.

Clauses affected:	⌘	6.6.2, 7.3.4, 7.4.2, 10
Other specs affected:	⌘	**X** Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

6.6.2 Generic Call Control Data Definitions

IpCall

Defines the address of an `IpCall` Interface.

IpCallRef

Defines a `Reference` to type IpCall.

IpAppCall

Defines the address of an `IpAppCall` Interface.

IpAppCallRef

Defines a `Reference` to type IpAppCall

IpAppCallRefRef

Defines a `Reference` to type IpAppCallRef.

TpCallIdentifierRef

Defines a `Reference` to type TpCallIdentifier.

TpCallIdentifier

Defines the `Sequence of Data Elements` that unambiguously specify the Generic Call object

Sequence Element Name	Sequence Element Type	Sequence Element Description
`CallReference`	`IpCallRef`	This element specifies the interface reference for the call object.
`CallSessionID`	`TpSessionID`	This element specifies the call session ID of the call.

IpAppCallControlManager

Defines the address of an `IpAppCallControlManager` Interface.

IpAppCallControlManagerRef

Defines a `Reference` to type IpAppCallControlManager.

IpCallControlManager

Defines the address of an `IpCallControlManager` Interface.

IpCallControlManagerRef

Defines a `Reference` to type IpCallControlManager.

TpCallAppInfo

Defines the `Tagged Choice of Data Elements` that specify application-related call information.

	Tag Element Type	
	`TpCallAppInfoType`	

Tag Element Value	Choice Element Type	Choice Element Name
`P_CALL_APP_ALERTING_MECHANISM`	TPCallAlertingMechanism	`CallAppAlertingMechanism`
`P_CALL_APP_NETWORK_ACCESS_TYPE`	TpCallNetworkAccessType	`CallAppNetworkAccessType`

P_CALL_APP_TELE_SERVICE	TpCallTeleService	CallAppTeleService
P_CALL_APP_BEARER_SERVICE	TpCallBearerService	CallAppBearerService
P_CALL_APP_PARTY_CATEGORY	TpCallPartyCategory	CallAppPartyCategory
P_CALL_APP_PRESENTATION_ADDRESS	TpAddress	CallAppPresentationAddress
P_CALL_APP_GENERIC_INFO	TpString	CallAppGenericInfo
P_CALL_APP_ADDITIONAL_ADDRESS	TpAddress	CallAppAdditionalAddress

TpCallAppInfoType

Defines the type of call application-related specific information.

Name	Value	Description
P_CALL_APP_UNDEFINED	0	Undefined
P_CALL_APP_ALERTING_MECHANISM	1	The alerting mechanism or pattern to use
P_CALL_APP_NETWORK_ACCESS_TYPE	2	The network access type (e.g. ISDN)
P_CALL_APP_TELE_SERVICE	3	Indicates the tele-service (e.g. telephony)
P_CALL_APP_BEARER_SERVICE	4	Indicates the bearer service (e.g. 64kbit/s unrestricted data).
P_CALL_APP_PARTY_CATEGORY	5	The category of the calling party
P_CALL_APP_PRESENTATION_ADDRESS	6	The address to be presented to other call parties
P_CALL_APP_GENERIC_INFO	7	Carries unspecified service-service information
P_CALL_APP_ADDITIONAL_ADDRESS	8	Indicates an additional address

TpCallAppInfoSet

Defines a Numbered Set of Data Elements of TpCallAppInfo.

TpCallChargePlan

Defines the Sequence of Data Elements that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
ChargeOrderType	TpCallChargeOrderCategory	Type of charging to be performed: time based charging or transparent charging or pre-defined charge plan.
ChargePerTime	TpChargePerTime	Charge per time. Only applicable when time based charging is selected.
TransparentCharge	TpOctetSet	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
ChargePlan	TpInt32	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when transparent charging is selected.
Currency	TpString	Currency unit according to ISO-4217:1995
AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
PartyToCharge	TpCallPartyToCharge	Party to be charged.

Valid Currencies are:

ADP, AED, AFA, ALL, AMD, ANG, AON, AOR, ARS, ATS, AUD, AWG, AZM, BAM,
BBD, BDT, BEF, BGL, BGN, BHD, BIF, BMD, BND, BOB, BOV, BRL, BSD, BTN,
BWP, BYB, BZD, CAD, CDF, CHF, CLF, CLP, CNY, COP, CRC, CUP, CVE, CYP,
CZK, DEM, DJF, DKK, DOP, DZD, ECS, ECV, EEK, EGP, ERN, ESP, ETB, EUR,
FIM, FJD, FKP, FRF, GBP, GEL, GHC, GIP, GMD, GNF, GRD, GTQ, GWP, GYD,
HKD, HNL, HRK, HTG, HUF, IDR, IEP, ILS, INR, IQD, IRR, ISK, ITL, JMD,
JOD, JPY, KES, KGS, KHR, KMF, KPW, KRW, KWD, KYD, KZT, LAK, LBP, LKR,
LRD, LSL, LTL, LUF, LVL, LYD, MAD, MDL, MGF, MKD, MMK, MNT, MOP, MRO,

MTL, MUR, MVR, MWK, MXN, MXV, MYR, MZM, NAD, NGN, NIO, NLG, NOK, NPR, NZD, OMR, PAB, PEN, PGK, PHP, PKR, PLN, PTE, PYG, QAR, ROL, RUB, RUR, RWF, SAR, SBD, SCR, SDD, SEK, SGD, SHP, SIT, SKK, SLL, SOS, SRG, STD, SVC, SYP, SZL, THB, TJR, TMM, TND, TOP, TPE, TRL, TTD, TWD, TZS, UAH, UGX, USD, USN, USS, UYU, UZS, VEB, VND, VUV, WST, XAF, XAG, XAU, XBA, XBB, XBC, XBD, XCD, XDR, XFO, XFU, XOF, XPD, XPF, XPT, XTS, XXX, YER, YUM, ZAL, ZAR, ZMK, ZRN, ZWD.

XXX is used for transactions where no currency is involved.

TpCallFault

Defines the cause of the call fault detected.

Name	Value	Description
P_CALL_FAULT_UNDEFINED	0	Undefined
P_CALL_TIMEOUT_ON_RELEASE	1	This fault occurs when the final report has been sent to the application, but the application did not explicitly release or deassign the call object, within a specified time. The timer value is operator specific.
P_CALL_TIMEOUT_ON_INTERRUPT	2	This fault occurs when the application did not instruct the gateway how to handle the call within a specified time, after the gateway reported an event that was requested by the application in interrupt mode. The timer value is operator specific.

TpCallPartyToCharge

Defines the party to be charged

Name	Value	Description
P_CALL_PARTY_ORIGINATING	0	Calling party, i.e. party that initiated the call. For application initiated calls this indicates that the first party requested to be in the call will be charged.
P_CALL_PARTY_DESTINATION	1	Called party, i.e. destination party

TpCallEndedReport

Defines the Sequence of Data Elements that specify the reason for the call ending.

Sequence Element Name	Sequence Element Type	Description
CallLegSessionID	TpSessionID	The leg that initiated the release of the call. If the call release was not initiated by the leg, then this value is set to –1.
Cause	TpCallReleaseCause	The cause of the call ending.

TpCallInfoReport

Defines the Sequence of Data Elements that specify the call information requested. Information that was not requested is invalid.

Sequence Element Name	Sequence Element Type	Description
CallInfoType	TpCallInfoType	The type of call report.
CallInitiationStartTime	TpDateAndTime	The time and date when the call, or follow-on call, was started.
CallConnectedToResourceTime	TpDateAndTime	The date and time when the call was connected to the resource. This data element is only valid when information on user interaction is reported.
CallConnectedToDestinationTime	TpDateAndTime	The date and time when the call 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. This data element is invalid when information on user interaction is reported with an intermediate report.

CallEndTime	TpDateAndTime	The date and time when the call or follow-on call or user interaction was terminated.
Cause	TpCallReleaseCause	The cause of the termination.

A callInfoReport will be generated at the end of user interaction and at the end of the connection with the associated address. This means that either the destination related information is present or the resource related information, but not both.

TpCallReleaseCause

Defines the Sequence of Data Elements that specify the cause of the release of a call.

Sequence Element Name	Sequence Element Type
Value	TpInt32
Location	TpInt32
NOTE: The Value and Location are specified as in ITU-T Recommendation Q.850.	

The following example was taken from Q.850 to aid understanding:

Equivalent Call Report	Cause Value Set by Application	Cause Value from Network
P_CALL_REPORT_BUSY	17	17
P_CALL_REPORT_NO_ANSWER	19	18,19,21
P_CALL_REPORT_DISCONNECT	16	16
P_CALL_REPORT_REDIRECTED	23	23
P_CALL_REPORT_SERVICE_CODE	31	NA
P_CALL_REPORT_ROUTING_FAILURE	3	Any other value

TpCallReport

Defines the Sequence of Data Elements that specify the call report and call leg report specific information.

Sequence Element Name	Sequence Element Type
MonitorMode	TpCallMonitorMode
CallEventTime	TpDateAndTime
CallReportType	TpCallReportType
AdditionalReportInfo	TpCallAdditionalReportInfo

TpCallAdditionalReportInfo

Defines the Tagged Choice of Data Elements that specify additional call report information for certain types of reports.

	Tag Element Type	
	TpCallReportType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_REPORT_UNDEFINED	NULL	Undefined
P_CALL_REPORT_PROGRESS	NULL	Undefined
P_CALL_REPORT_ALERTING	NULL	Undefined
P_CALL_REPORT_ANSWER	NULL	Undefined
P_CALL_REPORT_BUSY	TpCallReleaseCause	Busy
P_CALL_REPORT_NO_ANSWER	NULL	Undefined
P_CALL_REPORT_DISCONNECT	TpCallReleaseCause	CallDisconnect
P_CALL_REPORT_REDIRECTED	TpAddress	ForwardAddress
P_CALL_REPORT_SERVICE_CODE	TpCallServiceCode	ServiceCode
P_CALL_REPORT_ROUTING_FAILURE	TpCallReleaseCause	RoutingFailure

TpCallReportRequest

Defines the Sequence of Data Elements that specify the criteria relating to call report requests.

Sequence Element Name	Sequence Element Type
MonitorMode	TpCallMonitorMode
CallReportType	TpCallReportType
AdditionalReportCriteria	TpCallAdditionalReportCriteria

TpCallAdditionalReportCriteria

Defines the Tagged Choice of Data Elements that specify specific criteria.

	Tag Element Type	
	TpCallReportType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_REPORT_UNDEFINED	NULL	Undefined
P_CALL_REPORT_PROGRESS	NULL	Undefined
P_CALL_REPORT_ALERTING	NULL	Undefined
P_CALL_REPORT_ANSWER	NULL	Undefined
P_CALL_REPORT_BUSY	NULL	Undefined
P_CALL_REPORT_NO_ANSWER	TpDuration	NoAnswerDuration
P_CALL_REPORT_DISCONNECT	NULL	Undefined
P_CALL_REPORT_REDIRECTED	NULL	Undefined
P_CALL_REPORT_SERVICE_CODE	TpCallServiceCode	ServiceCode
P_CALL_REPORT_ROUTING_FAILURE	NULL	Undefined

TpCallReportRequestSet

Defines a Numbered Set of Data Elements of TpCallReportRequest.

TpCallReportType

Defines a specific call event report type.

Name	Value	Description
P_CALL_REPORT_UNDEFINED	0	Undefined.
P_CALL_REPORT_PROGRESS	1	Call routing progress event:an indication from the network that progress has been made in routing the call to the requested call party. This message may be sent more than once, or may not be sent at all by the gateway with respect to routing a given call leg to a given address.
P_CALL_REPORT_ALERTING	2	Call is alerting at the call party.
P_CALL_REPORT_ANSWER	3	Call answered at address.
P_CALL_REPORT_BUSY	4	Called address refused call due to busy.
P_CALL_REPORT_NO_ANSWER	5	No answer at called address.
P_CALL_REPORT_DISCONNECT	6	The media stream of the called party has disconnected. This does not imply that the call has ended. When the call is ended, the callEnded method is called. This event can occur both when the called party hangs up, or when the application explicitly releases the leg using IpCallLeg::release() This cannot occur when the app explicitly releases the call leg and the call.
P_CALL_REPORT_REDIRECTED	7	Call redirected to new address: an indication from the network that the call has been redirected to a new address.
P_CALL_REPORT_SERVICE_CODE	8	Mid-call service code received.
P_CALL_REPORT_ROUTING_FAILURE	9	Call routing failed - re-routing is possible.
P_CALL_REPORT_QUEUED	10	The call is being held in a queue. This event may be sent more than once during the routing of a call.

TpCallTreatment

Defines the Sequence of Data Elements that specify the the treatment for calls that will be handled only by the network (for example, call which are not admitted by the call load control mechanism).

Sequence Element Name	Sequence Element Type
ReleaseCause	TpCallReleaseCause
AdditionalTreatmentInfo	TpCallAdditionalTreatmentInfo

TpCallEventCriteriaResultSetRef

Defines a refernce to TpCallEventCriteriaResultSet.

TpCallEventCriteriaResultSet

Defines a set of TpCallEventCriteriaResult.

TpCallEventCriteriaResult

Defines a sequence of data elements that specify a requested call event notification criteria with the associated assignmentID.

Sequence Element Name	Sequence Element Type	Sequence Element Description
EventCriteria	TpCallEventCriteria	The event criteria that were specified by the application.
AssignmentID	TpInt32	The associated assignmentID. This can be used to disable the notification.

7.3.4 Interface Class IpAppMultiPartyCall

Inherits from: IpInterface

The Multi-Party call application interface is implemented by the client application developer and is used to handle call request responses and state reports.

<<Interface>>

IpAppMultiPartyCall

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

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

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

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

~~callFaultDetected (callSessionID : in TpSessionID, fault : in TpCallFault) : void~~

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

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

Method

getInfoRes()

This asynchronous method reports time information of the finished call or call attempt as well as release cause depending on which information has been requested by getInfoReq. This information may be used e.g. for charging purposes. The call information will possibly be sent after reporting of all cases where the call or a leg of the call has been disconnected or a routing failure has been encountered.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoReport : in TpCallInfoReport

Specifies the call information requested.

Method
getInfoErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method
superviseRes()

This asynchronous method reports a call supervision event to the application when it has indicated it's interest in these kind of events.

It is also called when the connection is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call

report : in TpCallSuperviseReport

Specifies the situation which triggered the sending of the call supervision response.

usedTime : in TpDuration

Specifies the used time for the call supervision (in milliseconds).

Method
superviseErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

~~*Method*~~
~~callFaultDetected()~~

~~This method indicates to the application that a fault in the network has been detected. The call may or may not have been terminated.~~

~~The system deletes the call object. Therefore, the application has no further control of call processing.~~

~~*Parameters*~~

~~**callSessionID : in TpSessionID**~~

~~Specifies the call session ID of the call in which the fault has been detected.~~

~~**fault : in TpCallFault**~~

~~Specifies the fault that has been detected.~~

Method

callEnded()

This method indicates to the application that the call has terminated in the network.

Note that the event that caused the call to end might have been received separately if the application was monitoring for it.

Parameters

callSessionID : in TpSessionID

Specifies the call sessionID.

report : in TpCallEndedReport

Specifies the reason the call is terminated.

Method

createAndRouteCallLegErr()

This asynchronous method 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, the network was unable to route the call, the parameters were incorrect, the request was refused, etc.). Note that the event cases that can be monitored and correspond to an unsuccessful setup of a connection (e.g. busy, no_answer) will be reported by eventReportRes() and not by this operation.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callLegReference : in TpCallLegIdentifier

Specifies the reference to the CallLeg interface that was created.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

7.4.2 State Transition Diagrams for IpMultiPartyCall

The state transition diagram shows the application view on the MultiParty Call object.

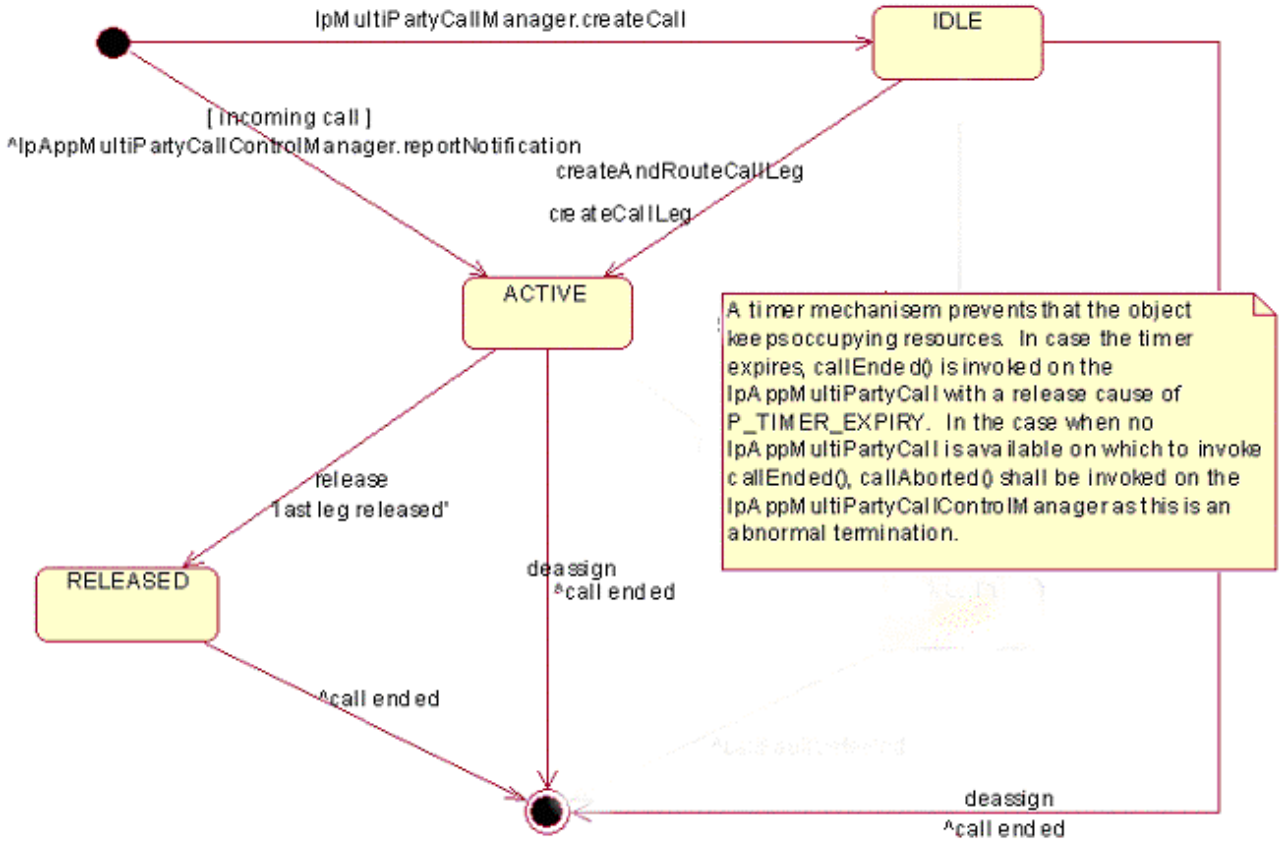


Figure : Application view on the MultiParty Call object

7.4.2.1 IDLE State

In this state the Call object has no Call Leg object associated to it.

The application can request for charging related information reports, call supervision, set the charge plan and set Advice Of Charge indicators. When the first Call Leg object is requested to be created a state transition is made to the Active state.

7.4.2.2 ACTIVE State

In this state the Call object has one or more Call Leg objects associated to it. The application is allowed to create additional Call Leg objects.

Furthermore, the application can request for call supervision. The Application can request charging related information reports, set the charge plan and set Advice Of Charge indicators in this state prior to call establishment.

~~7.4.2.3FAULTY State~~

~~A transition to this state is made when the Call object is in state IDLE and no requests from the application have been received during a certain period or when a non-recoverable fault was detected during the ACTIVE state.~~

~~In case the application requested for call related information previously, the application will be informed that this information is not available through getInfoError or SuperviseError and additionally the application is informed that the call object is transitioning to end state.~~

~~7.4.2.4~~7.4.2.3 RELEASED State

In this state the last Call leg object has released or the call itself was released. While the call is in this state, the requested call information will be collected and returned through getInfoReq() and / or superviseReq(). As soon as all information is returned, the application will be informed that the call has ended and Call object transition to the end state.

10 Common Call Control Data Types

TpCallAlertingMechanism

This data type is identical to a TpInt32, and defines the mechanism that will be used to alert a call party. The values of this data type are operator specific.

TpCallBearerService

This data type defines the type of call application-related specific information (Q.931: Information Transfer Capability, and 3G TS 22.002)

Name	Value	Description
P_CALL_BEARER_SERVICE_UNKNOWN	0	Bearer capability information unknown at this time
P_CALL_BEARER_SERVICE_SPEECH	1	Speech
P_CALL_BEARER_SERVICE_DIGITALUNRESTRICTED	2	Unrestricted digital information
P_CALL_BEARER_SERVICE_DIGITALRESTRICTED	3	Restricted digital information
P_CALL_BEARER_SERVICE_AUDIO	4	3.1 kHz audio
P_CALL_BEARER_SERVICE_ DIGITALUNRESTRICTEDTONES	5	Unrestricted digital information with tomes/announcements
P_CALL_BEARER_SERVICE_VIDEO	6	Video

TpCallChargePlan

Defines the Sequence of Data Elements that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
ChargeOrderType	TpCallChargeOrderCategory	Charge order
ChargePerTime	TpChargePerTime	Charge per time. Only applicable when time based charging is selected.
TransparentCharge	TpOctetSet	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
ChargePlan	TpInt32	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when transparent charging is selected.

Currency	TpString	Currency unit according to ISO-4217:1995
AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.

Valid Currencies are:

ADP, AED, AFA, ALL, AMD, ANG, AON, AOR, ARS, ATS, AUD, AWG, AZM, BAM, BBD, BDT, BEF, BGL, BGN, BHD, BIF, BMD, BND, BOB, BOV, BRL, BSD, BTN, BWP, BYB, BZD, CAD, CDF, CHF, CLF, CLP, CNY, COP, CRC, CUP, CVE, CYP, CZK, DEM, DJF, DKK, DOP, DZD, ECS, ECV, EEK, EGP, ERN, ESP, ETB, EUR, FIM, FJD, FKP, FRF, GBP, GEL, GHC, GIP, GMD, GNF, GRD, GTQ, GWP, GYD, HKD, HNL, HRK, HTG, HUF, IDR, IEP, ILS, INR, IQD, IRR, ISK, ITL, JMD, JOD, JPY, KES, KGS, KHR, KMF, KPW, KRW, KWD, KYD, KZT, LAK, LBP, LKR, LRD, LSL, LTL, LUF, LVL, LYD, MAD, MDL, MGF, MKD, MMK, MNT, MOP, MRO, MTL, MUR, MVR, MWK, MXN, MXV, MYR, MZM, NAD, NGN, NIO, NLG, NOK, NPR, NZD, OMR, PAB, PEN, PGK, PHP, PKR, PLN, PTE, PYG, QAR, ROL, RUB, RUR, RWF, SAR, SBD, SCR, SDD, SEK, SGD, SHP, SIT, SKK, SLL, SOS, SRG, STD, SVC, SYP, SZL, THB, TJR, TMM, TND, TOP, TPE, TRL, TTD, TWD, TZS, UAH, UGX, USD, USN, USS, UYU, UZS, VEB, VND, VUV, WST, XAF, XAG, XAU, XBA, XBB, XBC, XBD, XCD, XDR, XFO, XFU, XOF, XPD, XPF, XPT, XTS, XXX, YER, YUM, ZAL, ZAR, ZMK, ZRN, ZWD.

XXX is used for transactions where no currency is involved.

TpCallChargeOrder

Defines the Tagged Choice of Data Elements that specify the charge plan for the call.

	Tag Element Type	
	TpCallChargeOrderCategory	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_CHARGE_PER_TIME	TpChargePerTime	ChargePerTime
P_CALL_CHARGE_TRANSPARENT	TpOctetSet	TransparentCharge
P_CALL_CHARGE_PREDEFINED_SET	TpInt32	ChargePlan

TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
P_CALL_CHARGE_PER_TIME	0	Charge per time
P_CALL_CHARGE_TRANSPARENT	1	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records
P_CALL_CHARGE_PREDEFINED_SET	2	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user).

TpCallAdditionalChargePlanInfo

Defines the Tagged Choice of Data Elements that specify the charge plan for the call.

	Tag Element Type	
	TpCallChargeOrderCategory	

Tag Element Value	Choice Element Type	Choice Element Name	Description
P_CALL_CHARGE_PER_TIME	TpOctetSet	TimeAdditionalInfo	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
P_CALL_CHARGE_TRANSPARENT	NULL	Undefined	
P_CALL_CHARGE_PREDEFINED_SET	TpOctetSet	SetAdditionalInfo	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.

TpCallEndedReport

Defines the Sequence of Data Elements that specify the reason for the call ending.

Sequence Element Name	Sequence Element Type	Description
CallLegSessionID	TpSessionID	The leg that initiated the release of the call. If the call release was not initiated by the leg, then this value is set to –1.
Cause	TpCallReleaseCause	The cause of the call ending.

TpCallError

Defines the Sequence of Data Elements that specify the additional information relating to acall error.

Sequence Element Name	Sequence Element Type
ErrorTime	TpDateAndTime
ErrorType	TpCallErrorType
AdditionalErrorInfo	TpCallAdditionalErrorInfo

TpCallAdditionalErrorInfo

Defines the Tagged Choice of Data Elements that specify additional call error and call error specific information. This is also used to specify call leg errors and information errors.

	Tag Element Type	
	TpCallErrorType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_ERROR_UNDEFINED	NULL	Undefined
P_CALL_ERROR_INVALID_ADDRESS	TpAddressError	CallErrorInvalidAddress
P_CALL_ERROR_INVALID_STATE	NULL	Undefined

TpCallErrorType

Defines a specific call error.

Name	Value	Description
P_CALL_ERROR_UNDEFINED	0	Undefined; the method failed or was refused, but no specific reason can be given.
P_CALL_ERROR_INVALID_ADDRESS	1	The operation failed because an invalid address was given
P_CALL_ERROR_INVALID_STATE	2	The call was not in a valid state for the requested operation

~~TpCallFault~~

~~Defines the cause of the call fault detected.~~

~~Name~~	~~Value~~	~~Description~~
~~P_CALL_FAULT_UNDEFINED~~	~~0~~	~~Undefined~~
~~P_CALL_TIMEOUT_ON_RELEASE~~	~~1~~	~~This fault occurs when the final report has been sent to the application, but the application did not explicitly release or deassign the call object, within a specified time.~~ ~~The timer value is operator specific.~~
~~P_CALL_TIMEOUT_ON_INTERRUPT~~	~~2~~	~~This fault occurs when the application did not instruct the gateway how to handle the call within a specified time, after the gateway reported an event that was requested by the application in interrupt mode.~~ ~~The timer value is operator specific.~~

–TpCallInfoReport

Defines the Sequence of Data Elements that specify the call information requested. Information that was not requested is invalid.

Sequence Element Name	Sequence Element Type	Description
CallInfoType	TpCallInfoType	The type of call report.
CallInitiationStartTime	TpDateAndTime	The time and date when the call, or follow-on call, was started.
CallConnectedToResourceTime	TpDateAndTime	The date and time when the call was connected to the resource. This data element is only valid when information on user interaction is reported.
CallConnectedToDestinationTime	TpDateAndTime	The date and time when the call 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. This data element is invalid when information on user interaction is reported with an intermediate report.
CallEndTime	TpDateAndTime	The date and time when the call or follow-on call or user interaction was terminated.
Cause	TpCallReleaseCause	The cause of the termination.

A callInfoReport will be generated at the end of user interaction and at the end of the connection with the associated address. This means that either the destination related information is present or the resource related information, but not both.

TpCallInfoType

Defines the type of call information requested and reported. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_INFO_UNDEFINED	00h	Undefined
P_CALL_INFO_TIMES	01h	Relevant call times
P_CALL_INFO_RELEASE_CAUSE	02h	Call release cause
P_CALL_INFO_INTERMEDIATE	04h	Send only intermediate reports. When this is not specified the information report will only be sent when the call has ended. When intermediate reports are requested a report will be generated between follow-on calls, i.e., when a party leaves the call.

TpCallLoadControlMechanism

Defines the Tagged Choice of Data Elements that specify the applied mechanism and associated parameters.

	Tag Element Type	
	TpCallLoadControlMechanismType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_LOAD_CONTROL_PER_INTERVAL	TpCallLoadControlIntervalRate	CallLoadControlPerInterval

TpCallLoadControlIntervalRate

Defines the call admission rate of the call load control mechanism used. This data type indicates the interval (in milliseconds) between calls that are admitted.

Name	Value	Description
P_CALL_LOAD_CONTROL_ADMIT_NO_CALLS	0	Infinite interval (do not admit any calls)
	1 - 60000	Duration in milliseconds

TpCallLoadControlMechanismType

Defines the type of call load control mechanism to use.

Name	Value	Description
P_CALL_LOAD_CONTROL_PER_INTERVAL	1	admit one call per interval

TpCallMonitorMode

Defines the mode that the call will monitor for events, or the mode that the call is in following a detected event.

Name	Value	Description
P_CALL_MONITOR_MODE_INTERRUPT	0	The call event is intercepted by the call control service and call processing is interrupted. The application is notified of the event and call processing resumes following an appropriate API call or network event (such as a call release)
P_CALL_MONITOR_MODE_NOTIFY	1	The call event is detected by the call control service but not intercepted. The application is notified of the event and call processing continues
P_CALL_MONITOR_MODE_DO_NOT_MONITOR	2	Do not monitor for the event

TpCallNetworkAccessType

This data defines the bearer capabilities associated with the call. (3G TS 24.002) This information is network operator specific and may not always be available because there is no standard protocol to retrieve the information.

Name	Value	Description
P_CALL_NETWORK_ACCESS_TYPE_UNKNOWN	0	Network type information unknown at this time
P_CALL_NETWORK_ACCESS_TYPE_POT	1	POTS
P_CALL_NETWORK_ACCESS_TYPE_ISDN	2	ISDN
P_CALL_NETWORK_ACCESS_TYPE_DIALUPINTERNET	3	Dial-up Internet
P_CALL_NETWORK_ACCESS_TYPE_XDSL	4	xDLS
P_CALL_NETWORK_ACCESS_TYPE_WIRELESS	5	Wireless

TpCallPartyCategory

This data type defines the category of a calling party. (Q.763: Calling Party Category / Called Party Category)

Name	Value	Description
P_CALL_PARTY_CATEGORY_UNKNOWN	0	calling party's category unknown at this time
P_CALL_PARTY_CATEGORY_OPERATOR_F	1	operator, language French
P_CALL_PARTY_CATEGORY_OPERATOR_E	2	operator, language English
P_CALL_PARTY_CATEGORY_OPERATOR_G	3	operator, language German
P_CALL_PARTY_CATEGORY_OPERATOR_R	4	operator, language Russian
P_CALL_PARTY_CATEGORY_OPERATOR_S	5	operator, language Spanish
P_CALL_PARTY_CATEGORY_ORDINARY_SUB	6	ordinary calling subscriber
P_CALL_PARTY_CATEGORY_PRIORITY_SUB	7	calling subscriber with priority
P_CALL_PARTY_CATEGORY_DATA_CALL	8	data call (voice band data)
P_CALL_PARTY_CATEGORY_TEST_CALL	9	test call
P_CALL_PARTY_CATEGORY_PAYPHONE	10	payphone

TpCallServiceCode

Defines the Sequence of Data Elements that specify the service code and type of service code received during a call. The service code type defines how the value string should be interpreted.

Sequence Element Name	Sequence Element Type
CallServiceCodeType	TpCallServiceCodeType
ServiceCodeValue	TpString

TpCallServiceCodeType

Defines the different types of service codes that can be received during the call.

Name	Value	Description
P_CALL_SERVICE_CODE_UNDEFINED	0	The type of service code is unknown. The corresponding string is operator specific.
P_CALL_SERVICE_CODE_DIGITS	1	The user entered a digit sequence during the call. The corresponding string is an ascii representation of the received digits.
P_CALL_SERVICE_CODE_FACILITY	2	A facility information element is received. The corresponding string contains the facility information element as defined in ITU Q.932
P_CALL_SERVICE_CODE_U2U	3	A user-to-user message was received. The associated string contains the content of the user-to-user information element.
P_CALL_SERVICE_CODE_HOOKFLASH	4	The user performed a hookflash, optionally followed by some digits. The corresponding string is an ascii representation of the entered digits.

P_CALL_SERVICE_CODE_RECALL	5	The user pressed the register recall button, optionally followed by some digits. The corresponding string is an ascii representation of the entered digits.

TpCallSuperviseReport

Defines the responses from the call control service for calls that are supervised. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_SUPERVISE_TIMEOUT	01h	The call supervision timer has expired
P_CALL_SUPERVISE_CALL_ENDED	02h	The call has ended, either due to timer expiry or call party release. In case the called party disconnects but a follow-on call can still be made also this indication is used.
P_CALL_SUPERVISE_TONE_APPLIED	04h	A warning tone has been applied. This is only sent in combination with P_CALL_SUPERVISE_TIMEOUT
P_CALL_SUPERVISE_UI_FINISHED	0	The user interaction has finished.

TpCallSuperviseTreatment

Defines the treatment of the call by the call control service when the call supervision timer expires. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_SUPERVISE_RELEASE	01h	Release the call when the call supervision timer expires
P_CALL_SUPERVISE_RESPOND	02h	Notify the application when the call supervision timer expires
P_CALL_SUPERVISE_APPLY_TONE	04h	Send a warning tone to the originating party when the call supervision timer expires. If call release is requested, then the call will be released following the tone after an administered time period

TpCallTeleService

This data type defines the tele-service associated with the call. (Q.763: User Teleservice Information, Q.931: High Layer Compatitibility Information, and 3G TS 22.003)

Name	Value	Description
P_CALL_TELE_SERVICE_UNKNOWN	0	Teleservice information unknown at this time
P_CALL_TELE_SERVICE_TELEPHONY	1	Telephony
P_CALL_TELE_SERVICE_FAX_2_3	2	Facsimile Group 2/3
P_CALL_TELE_SERVICE_FAX_4_I	3	Facsimile Group 4, Class I
P_CALL_TELE_SERVICE_FAX_4_II_III	4	Facsimile Group 4, Classes II and III
P_CALL_TELE_SERVICE_VIDEOTEX_SYN	5	Syntax based Videotex
P_CALL_TELE_SERVICE_VIDEOTEX_INT	6	International Videotex interworking via gateways or interworking units
P_CALL_TELE_SERVICE_TELEX	7	Telex service
P_CALL_TELE_SERVICE_MHS	8	Message Handling Systems
P_CALL_TELE_SERVICE_OSI	9	OSI application
P_CALL_TELE_SERVICE_FTAM	10	FTAM application
P_CALL_TELE_SERVICE_VIDEO	11	Videotelephony
P_CALL_TELE_SERVICE_VIDEO_CONF	12	Videoconferencing
P_CALL_TELE_SERVICE_AUDIOGRAPH_CONF	13	Audiographic conferencing

P_CALL_TELE_SERVICE_MULTIMEDIA	14	Multimedia services
P_CALL_TELE_SERVICE_CS_INI_H221	15	Capability set of initial channel of H.221
P_CALL_TELE_SERVICE_CS_SUB_H221	16	Capability set of subsequent channel of H.221
P_CALL_TELE_SERVICE_CS_INI_CALL	17	Capability set of initial channel associated with an active 3.1 kHz audio or speech call.
P_CALL_TELE_SERVICE_DATATRAFFIC	18	Data traffic.
P_CALL_TELE_SERVICE_EMERGENCY_CALLS	1	Emergency Calls
P_CALL_TELE_SERVICE_SMS_MT_PP	2	Short message MT/PP
P_CALL_TELE_SERVICE_SMS_MO_PP	2	Short message MO/PP
P_CALL_TELE_SERVICE_CELL_BROADCAST	2	Cell Broadcast Service
P_CALL_TELE_SERVICE_ALT_SPEECH_FAX_3	2	Alternate speech and facsimile group 3
P_CALL_TELE_SERVICE_AUTOMATIC_FAX_3	2	Automatic Facsimile group 3
P_CALL_TELE_SERVICE_VOICE_GROUP_CALL	2	Voice Group Call Service
P_CALL_TELE_SERVICE_VOICE_BROADCAST	2	Voice Broadcast Service

TpCallTreatment

Defines the Sequence of Data Elements that specify the the treatment for calls that will be handled only by the network (for example, call which are not admitted by the call load control mechanism).

Sequence Element Name	Sequence Element Type
ReleaseCause	TpCallReleaseCause
AdditionalTreatmentInfo	TpCallAdditionalTreatmentInfo

TpCallTreatmentType

Defines the treatment for calls that will be handled only by the network.

Name	Value	Description
P_CALL_TREATMENT_DEFAULT	0	Default treatment
P_CALL_TREATMENT_RELEASE	1	Release the call
P_CALL_TREATMENT_SIAR	2	Send information to the user, and release the call (Send Info & Release)

TpCallAdditionalTreatmentInfo

Defines the Tagged Choice of Data Elements that specify the information to be sent to a call party.

	Tag Element Type	
	TpCallTreatmentType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_TREATMENT_DEFAULT	NULL	Undefined
P_CALL_TREATMENT_RELEASE	NULL	Undefined
P_CALL_TREATMENT_SIAR	TpUIInfo	InformationToSend

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010543**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **005** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Missing TpCallAppInfoSet description in OSA R4
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	The type description for TpCallAppInfoSet is missing in Section 7.6.2 of REL-4
Summary of change:	⌘	Added description for TpCallAppInfoSet in Section 7.6.2 of REL-4
Consequences if not approved:	⌘	If this description is not added REL-4 does not contain a specification for a type that is used

Clauses affected:	⌘	7.6.2
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	
Other comments:	⌘	

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

4.1.1 Multi-Party Call Control Data Definitions

IpCallLeg

Defines the address of an `IpCallLeg` Interface.

IpCallLegRef

Defines a `Reference` to type IpCallLeg.

IpCallLegRefRef

Defines a `Reference` to type IpCallLegRef.

IpAppCallLeg

Defines the address of an `IpAppCallLeg` Interface.

IpAppCallLegRef

Defines a `Reference` to type IpAppCallLeg.

IpMultiPartyCall

Defines the address of an `IpMultiPartyCall` Interface.

IpMultiPartyCallRef

Defines a `Reference` to type IpMultiPartyCall.

IpAppMultiPartyCall

Defines the address of an `IpAppMultiPartyCall` Interface.

IpAppMultiPartyCallRef

Defines a `Reference` to type IpAppMultiPartyCall.

IpMultiPartyCallControlManager

Defines the address of an `IpMultiPartyCallControlManager` Interface.

IpMultiPartyCallControlManagerRef

Defines a `Reference` to type IpMultiPartyCallControlManager.

IpAppMultiPartyCallControlManager

Defines the address of an `IpAppMultiPartyCallControlManager` Interface.

IpAppMultiPartyCallControlManagerRef

Defines a `Reference` to type IpAppMultiPartyCallControlManager..

TpAppCallLegRefSet

Defines `a Numbered Set of Data Elements` of IpAppCallLegRef.

IpAppCallLegRef

Defines a Reference to type IpAppCallLegRef.

IpAppMultiPartyCallRef

Defines a Reference to type IpAppMultiPartyCallRef.

TpMultiPartyCallIdentifier

Defines the Sequence of Data Elements that unambiguously specify the Call object

Sequence Element Name	Sequence Element Type	Sequence Element Description
CallReference	IpMultiPartyCallRef	This element specifies the interface reference for the Multi-party call object.
CallSessionID	TpSessionID	This element specifies the call session ID.

TpMultiPartyCallIdentifierRef

Defines a Reference to type TpMultiPartyCallIdentifier.

TpAppMultiPartyCallBack

Defines the Tagged Choice of Data Elements that references the application callback interfaces

	Tag Element Type	
	TpAppMultiPartyCallBackRefType	

Tag Element Value	Choice Element Type	Choice Element Name
P_APP_CALLBACK_UNDEFINED	NULL	Undefined
P_APP_MULTIPARTY_CALL_CALLBACK	IpAppMultiPartyCallRef	appMultiPartyCall
P_APP_CALL_LEG_CALLBACK	IpAppCallLegRef	appCallLeg
P_APP_CALL_AND_CALL_LEG_CALLBACK	TpAppCallLegCallBack	appMultiPartyCallAndCallLeg

TpAppMultiPartyCallBackRefType

Defines the type application call back interface.

Name	Value	Description
P_APP_CALLBACK_UNDEFINED	0	Application Call back interface undefined
P_APP_MULTIPARTY_CALL_CALLBACK	1	Application Multi-Party Call interface referenced
P_APP_CALL_LEG_CALLBACK	2	Application CallLeg interface referenced
P_APP_CALL_AND_CALL_LEG_CALLBACK	3	Application Multi-Party Call and CallLeg interface referenced

TpAppCallLegCallBack

Defines the Sequence of Data Elements that references a call and a call leg application interface.

Sequence Element Name	Sequence Element Type	
appMultiPartyCall	IpAppMultiPartyCallRef	
appCallLegSet	TpAppCallLegRefSet	Specifies the set of all call leg call back references. First in the set is the reference to the call back of the originating callLeg. In case there is a call back to a destination call leg this will be second in the set.

TpMultiPartyCallIdentifierSet

Defines a Numbered Set of Data Elements of TpMultiPartyCallIdentifier.

TpMultiPartyCallIdentifierSetRef

Defines a Reference to type TpMultiPartyCallIdentifierSet.

TpCallAppInfo

Defines the Tagged Choice of Data Elements that specify application-related call information.

	Tag Element Type	
	TpCallAppInfoType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_APP_ALERTING_MECHANISM	TPCallAlertingMechanism	CallAppAlertingMechanism
P_CALL_APP_NETWORK_ACCESS_TYPE	TpCallNetworkAccessType	CallAppNetworkAccessType
P_CALL_APP_TELE_SERVICE	TpCallTeleService	CallAppTeleService
P_CALL_APP_BEARER_SERVICE	TpCallBearerService	CallAppBearerService
P_CALL_APP_PARTY_CATEGORY	TpCallPartyCategory	CallAppPartyCategory
P_CALL_APP_PRESENTATION_ADDRESS	TpAddress	CallAppPresentationAddress
P_CALL_APP_GENERIC_INFO	TpString	CallAppGenericInfo
P_CALL_APP_ADDITIONAL_ADDRESS	TpAddress	CallAppAdditionalAddress
P_CALL_APP_ORIGINAL_DESTINATION_ADDRESS	TpAddress	CallAppOriginalDestinationAddress
P_CALL_APP_REDIRECTING_ADDRESS	TpAddress	CallAppRedirectingAddress

TpCallAppInfoSet

Defines a Numbered Set of Data Elements of TpCallAppInfo.

TpCallAppInfoType

Defines the type of call application-related specific information.

Name	Value	Description
P_CALL_APP_UNDEFINED	0	Undefined
P_CALL_APP_ALERTING_MECHANISM	1	The alerting mechanism or pattern to use
P_CALL_APP_NETWORK_ACCESS_TYPE	2	The network access type (e.g. ISDN)
P_CALL_APP_TELE_SERVICE	3	Indicates the tele-service (e.g. telephony)
P_CALL_APP_BEARER_SERVICE	4	Indicates the bearer service (e.g. 64 kbit/s unrestricted data).
P_CALL_APP_PARTY_CATEGORY	5	The category of the calling party
P_CALL_APP_PRESENTATION_ADDRESS	6	The address to be presented to other call parties
P_CALL_APP_GENERIC_INFO	7	Carries unspecified service-service information
P_CALL_APP_ADDITIONAL_ADDRESS	8	Indicates an additional address
P_CALL_APP_ORIGINAL_DESTINATION_ADDRESS	9	Contains the original address specified by the originating user when launching the call.
P_CALL_APP_REDIRECTING_ADDRESS	10	Contains the address of the user from which the call is diverting.

TpCallEventRequest

Defines the Sequence of Data Elements that specify the criteria relating to call report requests.

Sequence Element Name	Sequence Element Type
CallEventType	TpCallEventType
AdditionalCallEventCriteria	TpAdditionalCallEventCriteria
CallMonitorMode	TpCallMonitorMode

TpCallEventRequestSet

Defines a Numbered Set of Data Elements of TpCallEventRequest.

TpCallEventType

Defines a specific call event report type.

Name	Value	Description
P_CALL_EVENT_UNDEFINED	0	Undefined
P_CALL_EVENT_CALL_ATTEMPT	1	A Call attempt takes place (e.g. Off-hook event).
P_CALL_EVENT_ADDRESS_COLLECTED	2	The destination address has been collected.
P_CALL_EVENT_ADDRESS_ANALYSED	3	The destination address has been analysed.
P_CALL_EVENT_ALERTING	5	Call is alerting at the call party.
P_CALL_EVENT_ANSWER	6	Call answered at address.
P_CALL_EVENT_RELEASE	7	A Call has been released or the call could not be routed.
P_CALL_EVENT_REDIRECTED	8	Call redirected to new address: an indication from the network that the call has been redirected to a new address.
P_CALL_EVENT_SERVICE_CODE	9	Mid-call service code received.
P_CALL_EVENT_QUEUED	10	The Call Event has been queued. (no events are disarmed as a result of this)

The table below defines the disarming rules for dynamic events. In case such an event occurs the table shows which events are disarmed (are not monitored anymore) and should be re-armed by eventReportReq() in case the application is still interested in these events.

Event Occurred	Events Disarmed
P_CALL_EVENT_UNDEFINED	Not Applicable
P_CALL_EVENT_CALL_ATTEMPT	Not applicable, can only be armed as trigger
P_CALL_EVENT_ADDRESS_COLLECTED	P_CALL_EVENT_ADDRESS_COLLECTED
P_CALL_EVENT_ADDRESS_ANALYSED	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED
P_CALL_EVENT_PROGRESS	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED P_CALL_EVENT_PROGRESS
P_CALL_EVENT_ALERTING	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED P_CALL_EVENT_PROGRESS P_CALL_EVENT_ALERTING P_CALL_EVENT_RELEASE with criteria: P_USER_NOT_AVAILABLE P_BUSY P_NOT_REACHABLE P_ROUTING_FAILURE P_CALL_RESTRICTED P_UNAVAILABLE_RESOURCES
P_CALL_EVENT_ANSWER	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED P_CALL_EVENT_PROGRESS P_CALL_EVENT_ALERTING P_CALL_EVENT_RELEASE with criteria: P_USER_NOT_AVAILABLE P_BUSY P_NOT_REACHABLE P_ROUTING_FAILURE P_CALL_RESTRICTED P_UNAVAILABLE_RESOURCES P_NO_ANSWER P_PREMATURE_DISCONNECT P_CALL_EVENT_ANSWER
P_CALL_EVENT_RELEASE	All pending events are disarmed
P_CALL_EVENT_REDIRECTED	P_CALL_EVENT_REDIRECTED
P_CALL_EVENT_SERVICE_CODE	P_CALL_EVENT_SERVICE_CODE

TpAdditionalCallEventCriteria

Defines the Tagged Choice of Data Elements that specify specific criteria.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpInt32	MinAddressLength
P_CALL_EVENT_ADDRESS_ANALYSED	NULL	Undefined
P_CALL_EVENT_PROGRESS	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_RELEASE	TpCallReleaseCauseSet	ReleaseCauseSet
P_CALL_EVENT_REDIRECTED	NULL	Undefined

P_CALL_EVENT_SERVICE_CODE	TpCallServiceCode	ServiceCode

TpCallReleaseCauseSet

Defines a Numbered Set of Data Elements of TpCallReleaseCause.

TpCallEventInfo

Defines the Sequence of Data Elements that specify the event report specific information.

Sequence Element Name	Sequence Element Type
CallEventType	TpCallEventType
AdditionalCallEventInfo	TpCallAdditionalEventInfo
CallMonitorMode	TpCallMonitorMode
CallEventTime	TpDateAndTime

TpCallAdditionalEventInfo

Defines the Tagged Choice of Data Elements that specify additional call event information for certain types of events.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpAddress	CollectedAddress
P_CALL_EVENT_ADDRESS_ANALYSED	TpAddress	CalledAddress
P_CALL_EVENT_PROGRESS	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_RELEASE	TpCallReleaseCause	ReleaseCause
P_CALL_EVENT_REDIRECTED	TpAddress	ForwardAddress
P_CALL_EVENT_SERVICE_CODE	TpCallServiceCode	ServiceCode

TpCallNotificationRequest

Defines the Sequence of Data Elements that specify the criteria for an event notification

Sequence Element Name	Sequence Element Type	Description
CallNotificationScope	TpCallNoficationScope	Defines the scope of the notification request.
CallEventsRequested	TpCallEventRequestSet	Defines the events which are requested

TpCallNotificationScope

Defines a the sequence of Data elements that specify the scope of a notification request.

Of the addresses only the Plan and the AddrString are used for the purpose of matching the notifications against the criteria.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddressRange	Defines the destination address or address range for which the notification is requested.
OriginatingAddress	TpAddressRange	Defines the origination address or address range for which the notification is requested.
NotificationCallType	TpNotificationCallType	Defines wheter the notification is requested for a originating or terminating call.

TpNotificationCallType

Defines the type of call for which the notification is requested or reported.

Name	Value	Description
P_ORIGINATING	1	Indicates that the notification is related to the originating user in the call.
P_TERMINATING	2	Indicates that the notification is related to the terminating user in the call.

TpCallNotificationInfo

Defines the Sequence of Data Elements that specify the information returned to the application in a Call notification report.

Sequence Element Name	Sequence Element Type	Description
CallNotificationReportScope	TpCallNotificationReportScope	Defines the scope of the notification report.
CallAppInfo	TpCallAppInfoSet	Contains additonal call info.
CallEventInfo	TpCallEventInfo	Contains the event which is reported.

TpCallNotificationReportScope

Defines the Sequence of Data Elements that specify the scope for which a notification report was sent.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddress	Contains the destination address of the call.
OriginatingAddress	TpAddress	Contains the origination address of the call
NotificationCallType	TpNotificationCallType	Indicates if the notification was reported for an originating or terminating call.

TpNotificationRequested

Defines the Sequence of Data Elements that specify the criteria relating to event requests.

Sequence Element Name	Sequence Element Type
AppCallNotificationRequest	TpCallNotificationRequest
AssignmentID	TpInt32

TpNotificationsRequestedSet

Defines a numbered Set of Data Elements of TpNotificationRequested.

TpNotificationsRequestedSetRef

Defines a reference to the type TpNotificationsRequestSet.

TpCallReleaseCause

Defines the reason for which a call is released.

Name	Value	Description
P_UNDEFINED	0	The reason of release isn't known, because no info was received from the network.
P_USER_NOT_AVAILBLE	1	The user isn't available in the network. This means that the number isn't allocated or that the user isn't registered.
P_BUSY	2	The user is busy.
P_NO_ANSWER	3	No answer was received
P_NOT_REACHABLE	4	The user terminal isn't reachable
P_ROUTING_FAILURE	5	A routing failure occurred. For example an invalid address was received
P_PREMATURE_DISCONNECT	6	The user disconnected the call during setup phase.
P_DISCONNECTED	7	Call disconnect by the end user.
P_CALL_RESTRICTED	8	The call was subject of restrictions
P_UNAVAILABLE_RESOURCE	9	No resources where available to establisch the call.
P_GENERAL_FAILURE	10	A general network failure occurred.
P_TIMER_EXPIRY	11	The call was released because an activity timer expired.

TpCallLegIdentifier

Defines the Sequence of Data Elements that unambiguously specify the Call Leg object.

Sequence Element Name	Sequence Element Type	Sequence Element Description
CallLegReference	IpCallLegRef	This element specifies the interface reference for the callLeg object.
CallLegSessionID	TpSessionID	This element specifies the callLeg session ID.

TpCallLegIdentifierRef

Defines a Reference to type TpCallLegIdentifier.

TpCallLegIdentifierSet

Defines a Numbered Set of Data Elements of TpCallLegIdentifier.

TpCallLegIdentifierSetRef

Defines a Reference to type TpCallLegIdentifierSet.

TpCallLegAttachMechanism

Defines how a CallLeg should be attached to the call.

Name	Value	Description
P_CALLLEG_ATTACH_IMPLICITLY	0	CallLeg should be attached implicitly to the call.
P_CALLLEG_ATTACH_EXPLICITLY	1	CallLeg should be attached explicitly to the call by using the attachMedia() operation. This allows e.g. the application to do first user interaction to the party before he/she is placed in the call.

TpCallLegConnectionProperties

Defines the Sequence of Data Elements that specify the connection properties of the Call Leg object

Sequence Element Name	Sequence Element Type	Sequence Element Description
AttachMechanism	TpCallLegAttachMechanism	Defines how a CallLeg should be attached to the call.

TpCallLegInfoReport

Defines the Sequence of Data Elements that specify the call leg information requested.

Sequence Element Name	Sequence Element Type	Description
CallLegInfoType	TpCallLegInfoType	The type of the call leg.
CallLegStartTime	TpDateAndTime	The time and date when the call leg was started (i.e. the leg was routed).
CallLegConnectedToResourceTime	TpDateAndTime	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	The date and time 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. Either this element is valid or the CallConnectedToResourceTime is valid, depending on whether the report is sent as a result of user interaction.
CallLegEndTime	TpDateAndTime	The date and time when the call leg was released.
ConnectedAddress	TpAddress	The address of the party associated with the leg. If during the call the connected address was received from the party 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	TpCallReleaseCause	The cause of the termination. May be present with P_CALL_LEG_INFO_RELEASE_CAUSE was specified.
CallAppInfo	TpCallAppInfoSet	Additional information for the leg. May be present with P_CALL_LEG_INFO_APPINFO was specified.

TpCallLegInfoType

Defines the type of call leg information requested and reported. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_LEG_INFO_UNDEFINED	00h	Undefined
P_CALL_LEG_INFO_TIMES	01h	Relevant call times
P_CALL_LEG_INFO_RELEASE_CAUSE	02h	Call leg release cause
P_CALL_LEG_INFO_ADDRESS	04h	Call leg connected address
P_CALL_LEG_INFO_APPINFO	08h	Call leg application related information

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010612**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **006** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Redirecting a call leg vs. creating a call leg clarification in OSA R4
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The methods createAndRouteCallLegReq and routeReq do not specify the difference between creating a call leg and redirecting a call leg in REL-4
Summary of change:	⌘	Added description for createAndRouteCallLegReq and routeReq of REL-4
Consequences if not approved:	⌘	If these descriptions are not accepted, there is no means to distinguish between creating a call leg and redirecting a call leg.

Clauses affected:	⌘	7.3.3, 7.3.5
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	
Other comments:	⌘	

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

7.3.3 Interface Class IpMultiPartyCall

Inherits from: IpService

The Multi-Party Call provides the possibility to control the call routing, to request information from the call, control the charging of the call, to release the call and to supervise the call. It also gives the possibility to manage call legs explicitly. An application may create more then one call leg.

<<Interface>> IpMultiPartyCall
getCallLegs (callSessionID : in TpSessionID) : TpCallLegIdentifierSet createCallLeg (callSessionID : in TpSessionID, appCallLeg : in IpAppCallLegRef) : TpCallLegIdentifier createAndRouteCallLegReq (callSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet, targetAddress : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, appLegInterface : in IpAppCallLegRef) : TpCallLegIdentifier release (callSessionID : in TpSessionID, cause : in TpCallReleaseCause) : void deassignCall (callSessionID : in TpSessionID) : void getInfoReq (callSessionID : in TpSessionID, callInfoRequested : in TpCallInfoType) : void setChargePlan (callSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : void setAdviceOfCharge (callSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : void superviseReq (callSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : void

Method

getCallLegs()

This method requests the identification of the call leg objects associated with the call object. Returns the legs in the order of creation.

Returns callLegList: Specifies the call legs associated with the call. The set contains both the sessionIDs and the interface references.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Returns

TpCallLegIdentifierSet

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

createCallLeg()

This method requests the creation of a new call leg object.

Returns callLeg: Specifies the interface and sessionID of the call leg created.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

appCallLeg : in IpAppCallLegRef

Specifies the application interface for callbacks from the call leg created.

Returns

TpCallLegIdentifier

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

Method

createAndRouteCallLegReq()

This asynchronous operation requests creation and routing of a new callLeg. In case the connection to the destination party is established successfully the CallLeg is attached to the call, i.e. no explicit setMedia() operation is needed. Requested events will be reported on the IpAppCallLeg interface. This interface the application must provide through the appLegInterface parameter.

The extra address information such as originatingAddress is optional. If not present (i.e., the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in corresponding addresses from the route is used, otherwise the network or gateway provided numbers will be used.

If the application wishes that the call leg should be represented in the network as being a redirection it should include a value for the field P_CALL_APP_ORIGINAL_DESTINATION_ADDRESS of TpCallAppInfo.

If this method in invoked, and call reports have been requested, yet the IpAppCallLeg interface parameter is NULL, this method shall throw the P_NO_CALLBACK_ADDRESS_SET exception.

Returns callLegReference: Specifies the reference to the CallLeg interface that was created.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

eventsRequested : in TpCallEventRequestSet

Specifies the event specific criteria used by the application to define the events required. Only events that meet these criteria are reported. Examples of events are "adress analysed", "answer", "release".

targetAddress : in TpAddress

Specifies the destination party to which the call should be routed.

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call (such as alerting method, tele-service type, service identities and interaction indicators).

appLegInterface : in IpAppCallLegRef

Specifies a reference to the application interface that implements the callback interface for the new call leg. Requested events will be reported by the eventReportRes() operation on this interface.

Returns

TpCallLegIdentifier

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_ADDRESS , P_UNSUPPORTED_ADDRESS_PLAN, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA

Method

release()

This method requests the release of the call object and associated objects. The call will also be terminated in the network. If the application requested reports to be sent at the end of the call (e.g., by means of getInfoReq) these reports will still be sent to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

deassignCall()

This method requests 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 call information reports, call leg event reports or call Leg information reports requested, then these reports will be disabled and any related information discarded.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

getInfoReq()

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method must be invoked before the call is routed to a target address. Two types of reports can be requested; a final report or intermediate reports.

A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.

Intermediate reports are received when the destination leg or party terminates or when the call ends.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType

Specifies the call information that is requested.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setChargePlan()

Set an operator specific charge plan for the call. The charge plan must be set before the call is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tariffSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

superviseReq()

The application calls this method to supervise a call. The application can set a granted connection time for this call. If an application calls this operation before it routes a call or a user interaction operation the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

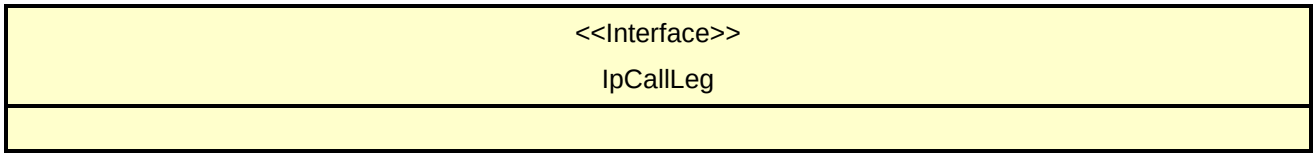
Specifies how the network should react after the granted connection time expired.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

7.3.5 Interface Class IpCallLeg

Inherits from: The call leg interface represents the logical call leg associating a call with an address. The call leg tracks its own states and allows charging summaries to be accessed. The leg represents the signalling relationship between the call and an address. An application that uses the IpCallLeg interface to set up connections has more control, e.g. by defining leg specific event request and can obtain call leg specific report and events.



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

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

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

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

getCall (callLegSessionID : in TpSessionID) : TpMultiPartyCallIdentifier

attachMedia (callLegSessionID : in TpSessionID) : void

detachMedia (callLegSessionID : in TpSessionID) : void

getLastRedirectedAddress (callLegSessionID : in TpSessionID) : TpAddress

continueProcessing (callLegSessionID : in TpSessionID) : void

getMoreDialledDigitsReq (callLegSessionID : in TpSessionID, length : in TpInt32) : void

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

setAdviceOfCharge (callLegSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tarrifSwitch : in TpDuration) : void

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

deassign (callLegSessionID : in TpSessionID) : void

Method

routeReq()

This asynchronous method requests routing of the call leg to the remote party indicated by the targetAddress.

In case the connection to the destination party is established successfully the CallLeg will be either detached or attached to the call based on the attach Mechanism values specified in the connectionProperties parameter.

The extra address information such as originatingAddress is optional. If not present (i.e. the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in the corresponding addresses from the route is used, otherwise network or gateway provided addresses will be used.

If the application wishes that the call leg should be represented in the network as being a redirection it should include a value for the field P_CALL_APP_ORIGINAL_DESTINATION_ADDRESS of TpCallAppInfo.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

targetAddess : in TpAddress

Specifies the destination party to which the call leg should be routed

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call leg (such as alerting method, tele-service type, service identities and interaction indicators).

connectionProperties : in TpCallLegConnectionProperties

Specifies the properties of the connection.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

eventReportReq()

This asynchronous method sets, clears or changes the criteria for the events that the call leg object will be set to observe.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

eventsRequested : in TpCallEventRequestSet

Specifies the event specific criteria used by the application to define the events required. Only events that meet these criteria are reported. Examples of events are "address analysed", "answer", "release".

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_EVENT_TYPE, P_INVALID_CRITERIA

Method

release()

This method requests the release of the call leg. If successful, the associated address (party) will be released from the call, and the call leg deleted. Note that in some cases releasing the party may lead to release of the complete call in the network. The application will be informed of this with callEnded().

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

getInfoReq()

This asynchronous method requests information associated with the call leg to be provided at the appropriate time (for example, to calculate charging). Note: in the call leg information must be accessible before the objects of concern are deleted.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

callLegInfoRequested : in TpCallLegInfoType

Specifies the call leg information that is requested.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

getCall()

This method requests the call associated with this call leg.

Returns callReference:Specifies the interface and sessionID of the call associated with this call leg.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Returns

TpMultiPartyCallIdentifier

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

attachMedia()

This method requests that the call leg be attached to its call object. This will allow transmission on all associated bearer connections or media channels to and from other parties in the call. The call leg must be in the connected state for this method to complete successfully.

Parameters

callLegSessionID : in TpSessionID

Specifies the sessionID of the call leg to attach to the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

detachMedia()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the sessionID of the call leg to detach from the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

getLastRedirectedAddress()

Queries the last address the leg has been redirected to.

Returns redirectedAddress: Specifies the last address where the call leg was redirected to.

Parameters

callLegSessionID : in TpSessionID

Specifies the call session ID of the call leg.

Returns

TpAddress

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

continueProcessing()

This operation continues processing of the call leg. Applications can invoke this operation after call leg processing was interrupted due to detection of a notification or event the application subscribed it's interest in.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

getMoreDialledDigitsReq()

This asynchronous method requests to collect further digits and return them to the application. Depending on the administered data, the network may indicate a new call to the gateway if a caller goes off-hook or dialled only a few digits. The application then gets a new call event which contains no digits or only the few dialled digits in the event data. The application should then use this method if it requires more dialled digits, e.g. to perform screening.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call.

length : in TpInt32

Specifies the maximum number of digits to collect.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setChargePlan()

Set an operator specific charge plan for the cal leg. The charge plan must be set before the call leg is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tarrifSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

superviseReq()

The application calls this method to supervise a call leg. The application can set a granted connection time for this call. If an application calls this function before it calls a routeReq() or a user interaction function the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

Specifies how the network should react after the granted connection time expired.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

deassign()

This method requests that the relationship between the application and the call leg and associated objects be de-assigned. It leaves the call leg 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 leg information reports requested, then these reports will be disabled and any related information discarded.

The application should always either release or deassign the call leg when it is finished with the call, leg unless callFaultDetected is received by the application.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010614**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **007** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	The behaviour of the MPCC API is unclear. The Call Leg STDs for MPCC are needed together with the proposed changes for the associated methods and data.
Summary of change:	⌘	The following main changes apply 1) Introduction of Originating Call Leg STD and Terminating Call Leg STD 2) Changed semantics and modifications to existing MPCC methods to clarify behaviour of the MPCC API. 3) Removal of method getMoreDialledDigitsReq\Res since a detailed behaviour is undefined and requested functionality for digit collection is already covered with existing methods eventReportReq\Res. 4) Modifications for MPCC data types to correct errors and adapt to the clarified behaviour of the MPCC API supporting the new Originating Call Leg STD and Terminating Call Leg STDs. This also covers the separation of events into originating and terminating events for: call attempt, call attempt authorized, mid-call (service code), and clarifications for event handling.
Consequences if not approved:	⌘	Lacking clarification of the behaviour of the MPCC API if the Call LEG STDs and associated modifications are not introduced.

Clauses affected:	⌘	7.3.5, 7.3.6, 7.4.2, 7.4.2.5, 7.4.3, 7.4.3.1 to 7.4.3.10, 7.6 and 7.6.2		
Other specs affected:	⌘	☒ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘			

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

7.3.5 Interface Class IpCallLeg

~~Inherits from:~~ The call leg interface represents the logical call leg associating a call with an address. The call leg tracks its own states and allows charging summaries to be accessed. The leg represents the signalling relationship between the call and an address. An application that uses the IpCallLeg interface to set up connections has good~~more~~ control, e.g. by defining leg specific event request and can obtain call leg specific report and events.

<<Interface>> IpCallLeg
routeReq (callLegSessionID : in TpSessionID, targetAddess : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, connectionProperties : in TpCallLegConnectionProperties) : void eventReportReq (callLegSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet) : void release (callLegSessionID : in TpSessionID, cause : in TpCallReleaseCause) : void getInfoReq (callLegSessionID : in TpSessionID, callLegInfoRequested : in TpCallLegInfoType) : void getCall (callLegSessionID : in TpSessionID) : TpMultiPartyCallIdentifier attachMedia (callLegSessionID : in TpSessionID) : void detachMedia (callLegSessionID : in TpSessionID) : void getLastRedirectedAddress (callLegSessionID : in TpSessionID) : TpAddress continueProcessing (callLegSessionID : in TpSessionID) : void ~~getMoreDialledDigitsReq (callLegSessionID : in TpSessionID, length : in TpInt32) : void~~ setChargePlan (callLegSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : void setAdviceOfCharge (callLegSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tarrifSwitch : in TpDuration) : void superviseReq (callLegSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : void deassign (callLegSessionID : in TpSessionID) : void

Method
routeReq()

This asynchronous method requests routing of the call leg to the remote party indicated by the targetAddress.

In case the connection to the destination party is established successfully the CallLeg will be either detached or attached to the call based on the attach Mechanism values specified in the connectionProperties parameter.

The extra address information such as originatingAddress is optional. If not present (i.e. the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in the corresponding addresses from the route is used, otherwise network or gateway provided addresses will be used. This operation continues processing of the call leg.

…

Method
release()

This method requests the release of the call leg. If successful, the associated address (party) will be released from the call, and the call leg deleted. Note that in some cases releasing the party may lead to release of the complete call in the network. The application will be informed of this with callEnded().This operation continues processing of the call leg

…

Method
getLastRedirectedAddress()

Queries the last address the leg has been redirected to. . If this method is invoked on the Originating Call Leg, exception P_INVALID_STATE will be thrown.

Parameters

callLegSessionID : in TpSessionID

Specifies the call session ID of the call leg.

redirectedAddress : out TpAddressRef

Specifies the last address where the call leg was redirected to.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_STATE

…

Method
attachMedia()

This method requests that the call leg be attached to its call object. This will allow transmission on all associated bearer connections or media streams~~channels~~ to and from other parties in the call. The call leg must be in the connected state for this method to complete successfully.

Parameters

callLegSessionID : in TpSessionID

Specifies the sessionID of the call leg to attach to the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method
detachMedia()

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

…

Method
continueProcessing()

This operation continues processing of the call leg. Applications can invoke this operation after call leg processing was interrupted due to detection of a notification or event the application subscribed its~~it's~~ interest in.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

~~*Method*~~

~~getMoreDialledDigitsReq()~~

~~This asynchronous method requests to collect further digits and return them to the application. Depending on the administered data, the network may indicate a new call to the gateway if a caller goes off-hook or dialled only a few digits. The application then gets a new call event which contains no digits or only the few dialled digits in the event data. The application should then use this method if it requires more dialled digits, e.g. to perform screening.~~

~~*Parameters*~~

~~**callLegSessionID : in TpSessionID**~~

~~Specifies the call leg session ID of the call.~~

~~**length : in TpInt32**~~

~~Specifies the maximum number of digits to collect.~~

~~*Raises*~~

~~**TpCommonExceptions, P_INVALID_SESSION_ID**~~

Method

deassign()

This method requests that the relationship between the application and the call leg- and associated objects be de-assigned. It leaves the call leg 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 leg information reports requested, then these reports will be disabled and any related information discarded.

The application should not~~always either~~ release or deassign the call leg when it received a callLegEnded() or callEnded()~~is finished with the call, leg- unless callFaultDetected is received by the application~~. This operation continues processing of the call leg.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

7.3.6 Interface Class IpAppCallLeg

Inherits from: IpInterface

IpService

The application call leg interface is implemented by the client application developer and is used to handle responses and errors associated with requests on the call leg in order to be able to receive leg specific information and events.

<<Interface>> IpAppCallLeg
eventReportRes (callLegSessionID : in TpSessionID, eventInfo : in TpCallEventInfo) : TpResult eventReportErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult getInfoRes (callLegSessionID : in TpSessionID, callLegInfoReport : in TpCallLegInfoReport) : TpResult getInfoErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult routeErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult ~~getMoreDialledDigitsRes (callSessionID : in TpSessionID, digits : in TpString) : TpResult~~ ~~getMoreDialledDigitsErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult~~ superviseRes (callLegSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : TpResult superviseErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult callLegEnded~~connectionEnded~~ (callLegSessionID : in TpSessionID, cause : in TpCallReleaseCause) : TpResult

…

~~*Method*~~

~~getMoreDialledDigitsRes()~~

~~This asynchronous method returns the collected digits to the application.~~

~~*Parameters*~~

~~callSessionID : in TpSessionID~~

~~Specifies the call session ID of the call.~~

~~digits : in TpString~~

~~Specifies the additional dialled digits if the string length is greater than zero.~~

~~*Method*~~

~~getMoreDialledDigitsErr()~~

~~This asynchronous method reports an error in collecting digits to the application.~~

~~*Parameters*~~

~~**callSessionID : in TpSessionID**~~

~~Specifies the call session ID of the call.~~

~~**errorIndication : in TpCallError**~~

~~Specifies the error which led to the original request failing.~~

…

Method

callLegEnded~~connectionEnded~~()

This method indicates to the application that the~~-~~ leg ~~connection~~ has terminated ~~in the network. However, t~~. The application ~~may still~~has received all requested ~~some~~ results (e.g., getInfoRes) related to the call leg. The call leg will be destroyed after returning from this method. ~~The application is expected to deassign the call leg object after having received the connectionEnded.~~

~~Note that the event that caused the connection to end might also be received separately if the application was monitoring for it.~~

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpCallReleaseCause

Method

eventReportRes()

This asynchronous method reports that an event has occurred that was requested to be reported (for example, a mid-call event, the party has requested to disconnect, etc.).

Depending on the type of event received, outstanding requests for events are discarded. The exact details of these so-called disarming rules are captured in the data definition of

the event type.

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of P_TIMER_EXPIRY.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg on which the event was detected.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

7.4.2 State Transition Diagram for IpMultiPartyCall

…

7.4.2.5 Overview of allowed methods

Methods applicable	Call Control Call State	Call Control Manager State	~~Call Control Call Leg state~~
getCallLegs,	Idle, Active, Released	-	
createCallLeg~~s~~, createAndRouteCallLegReq, setAdviceOfCharge, superviseReq,	Idle, Active	Active	
Release	Active	Active	
Deassign	Idle, Active	-	
GetInfoReq	Idle	Active	
SetChargePlan	Idle, Active	Active	~~Alerting, Connected~~

7.4.3 State Transition Diagrams for IpCallLeg

The IpCallLeg State Transition Diagram is divided in two State Transition Diagrams, one for the originating call leg and one for the terminating call leg.

Call Leg State Model General Objectives:

1 Events in backwards direction (upstream), coming from terminating leg, are not visible in originating leg model.

2 Events in forwards direction (downstream), coming from originating leg, are not visible in terminating leg model.

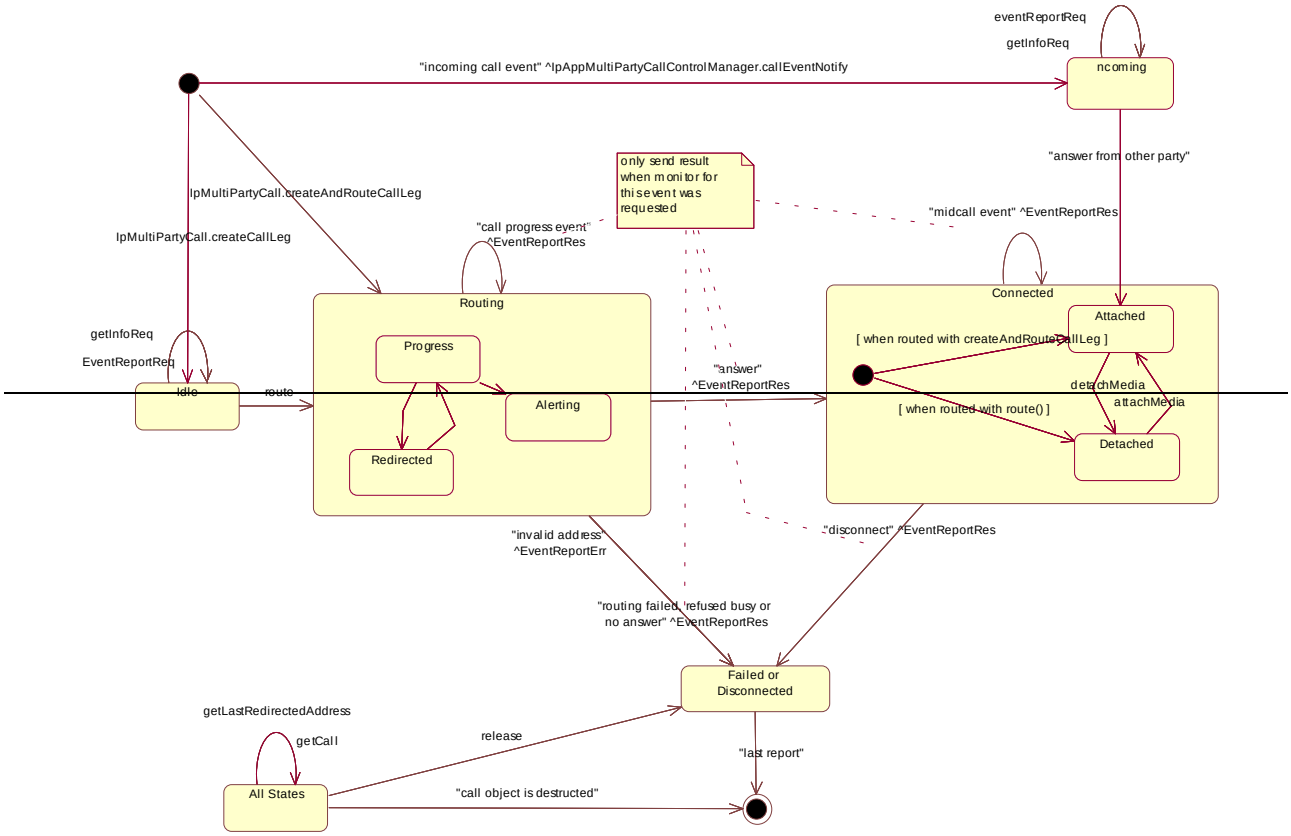
3 States are as seen from the application: if there is no change in the method an application is permitted to apply on the IpCallLeg object, then there is no state change. Therefore receipt of e.g. answer or alerting events on terminating leg do not change state. NOTE 2

4 The application is to send a request to continue processing (using an appropriate method like continueProcessing) for each leg and event reported in monitor mode 'interrupt'. The call processing is resumed in the network when no leg in the call is left suspended.

5 In case on a leg more than one network event (for example mid-call event 'service_code') is to be reported to the application at quasi the same time, then the events are to be reported one by one to the application in the order received from the network. When for a leg an event is reported in interrupt mode, a next pending event is not to be reported to the application until a request to resume call processing for the current reported event has been received on the leg.

NOTE1: Call processing is suspended if for a leg a network event is met, which was requested to be monitored in the P_CALL_MONITOR_MODE_INTERRUPT.

NOTE2: Even though there in the Originating Call Leg STD is no change in the methods the application is permitted to apply to the IpCallLeg object for the states Analysing and Active, separate states are maintained. The states may therefore from an application viewpoint appear as just one state that may be have substates like Analysing and Active. The digit collection task in state Analysing state may be viewed as a specialised task that may not at all be applicable in some networks and therefore here described as being a state on its own.



1.

~~Figure : Application view on the CallLeg object~~

7.4.3.1 Originating Call Leg

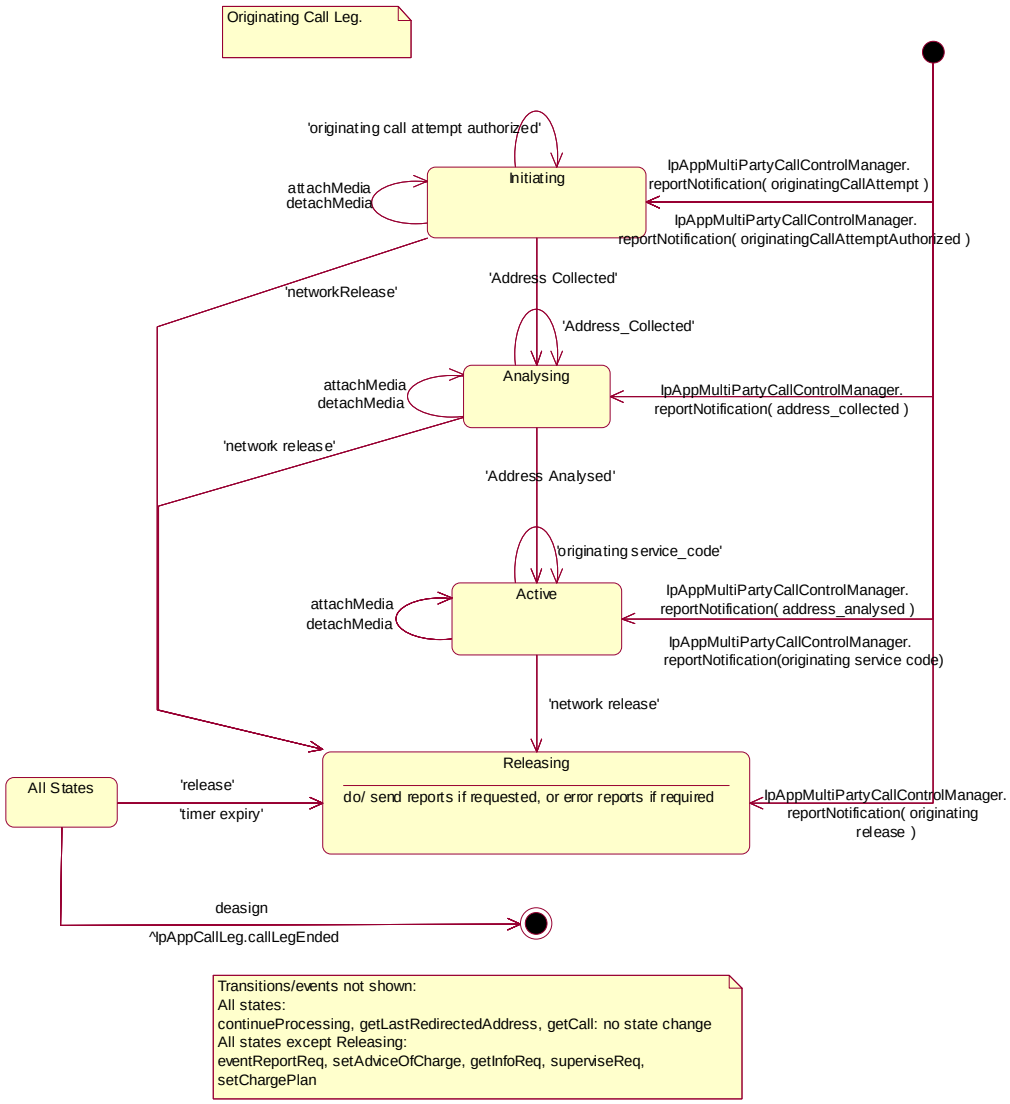


Figure : Application view on the Originating CallLeg object

7.4.3.1.1 Initiating

Entry events:

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Originating_Call_Attempt" initial notification ~~trigger~~ criterion.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Originating_Call_Attempt_Authorised" initial notification~~trigger~~ criterion.

Functions:

In this state the network checks the authority/ability of the party to place the connection to the remote (destination) party with the given properties, e.g. based on the originating party's identity and service profile.
The setup of the connection for the party has been initiated and the application activity timer is being provided.

The figure below shows the order in which network events may be detected in the Initiating state and depending on the monitor mode be reported to the application.

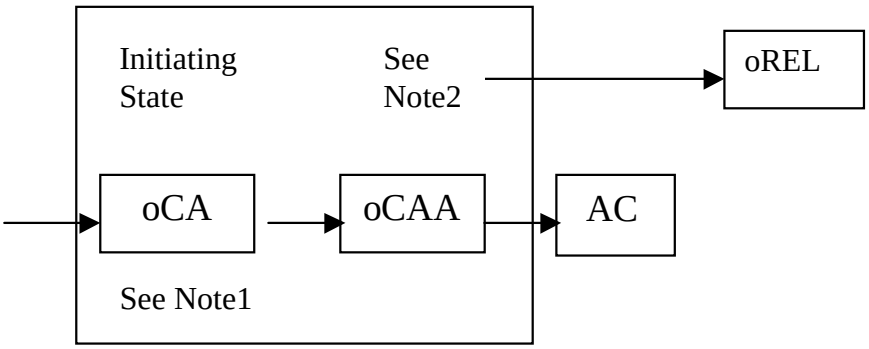


Figure : Application view on event reporting order in Initiating State

Note 1: Event oCA only applicable as an intitial notification .
Note 2: The release event (oREL) can occur in any state resulting in a transition to Releasing state.

Abbreviations used for the events:

oCA Originating Call Attempt; oCAA Originating Call Attempt Authorized; AC Address Collected, oREL Originating Release.

In this state the following functions are applicable:

- The detection of a "Originating_Call_Attempt" initial notification criterion.

- The detection of an "Originating_Call_Attempt_Authorised" initial notification~~r~~ criterion as a result that the call attempt authorisation is successful.

- The report of the "Originating_Call_Attempt_Authorised" event indication whereby the following functions are performed:

  i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_CALL_ATTEMPT_AUTHORISED then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_CALL_ATTEMPT_AUTHORISED then the event is notified and call leg processing continues.

iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_CALL_ATTEMPT_AUTHORISED then no monitoring is performed.

- The receipt of destination address information, i.e. initial information package/dialling string as received from calling party.

- Resumption of suspended call leg processing occurs on receipt of a continueProcessing() method.

Exit events:

- Availability of destination address information, i.e. the initial information package/dialling string received from the calling party.

- Application activity timer expiry indicating that no requests from the application have been received during a certain period.

- Receipt of a deassign() method.

- Receipt of a release() method.

- Detection of a "originating release" indication as a result of a premature disconnect from the calling party.

7.4.3.1.2 Analysing

Entry events:

- Availability of an "Address_Collected" event indication as a result of the receipt of the (complete) initial information package/dialling string from the calling party.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Address_Collected" initial notification criterion.

Functions:

In this state the destination address provided by the calling party is collected and analysed.
The received information (dialled address string from the calling party) is being collected and examined in accordance to the dialling plan in order to determine end of address information (digit) collection. Additional address digits can be collected. Upon completion of address collection the address is analysed.
The address analysis is being made according to the dialling plan in force to determine the routing address of the call leg connection and the connection type (e.g. local, transit, gateway).

The request (with eventReportReq method) to collect a variable number of more address digits and report them to the application (within eventReportRes method)) is handled within this state. The collection of more digits as requested and the reporting of received digits to the application (when the digit collect criteria is met) is done in this state. This action is recursive, e.g. the application could ask for 3 digits to be collected and when report request can be done repeatedly, e.g. the application may for example request first for 3 digits to be collected and when reported request further digits.

The figure below shows the order in which network events may be detected in the Analysing state and depending on the monitor mode be reported to the application.

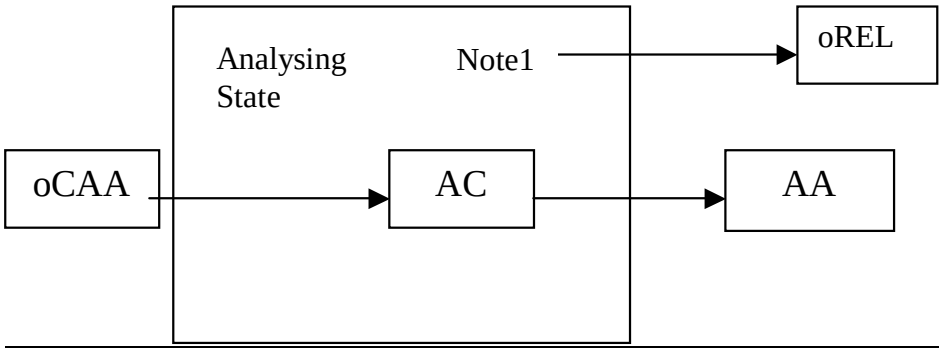


Figure : Application view on event reporting order in Analysing State

Note 1: The release event (oREL) can occur in any state resulting in a transition to Releasing state.

Abbreviations used for the events:

oCAA Originating Call Attempt Authorized; AC Address Collected; AA Address Analysed; oREL Originating Release.

In this state the following functions are applicable:

- The detection of a "Address_Collected" initial notification criterion.
- On receipt of the "Address_Collected" indication the following functions are performed:
  i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_ADDRESS_COLLECTED then the event is intercepted and call leg processing is suspended.
  ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_ADDRESS_COLLECTED then the event is notified and call leg processing continues.
  iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_ADDRESS_COLLECTED then no monitoring is performed.
- Receipt of a eventReportReq() method defining the criteria for the events the call leg object is to observe.
- Resumption of suspended call leg processing occurs on receipt of a continueProcessing() or a routeReq() method.

Exit events:

- Detection of an "Address_Analysed" indication as a result of the availability of the routing address and nature of address.
- Receipt of a deassign() method.
- Receipt of a release() method.
- Detection of a "originating release" indication as a result of a premature disconnect from the calling party.

7.4.3.1.3 Active

Entry events:

- Receipt of an "Address_Analysed" indication as a result of the availability of the routing address and nature of address.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Address_Analysed initial indication criterion.

Functions:

In this state the call leg connection to the calling party exists and originating mid call events can be received.

The figure below shows the order in which network events may be detected in the Active state and depending on the monitor mode be reported to the application.

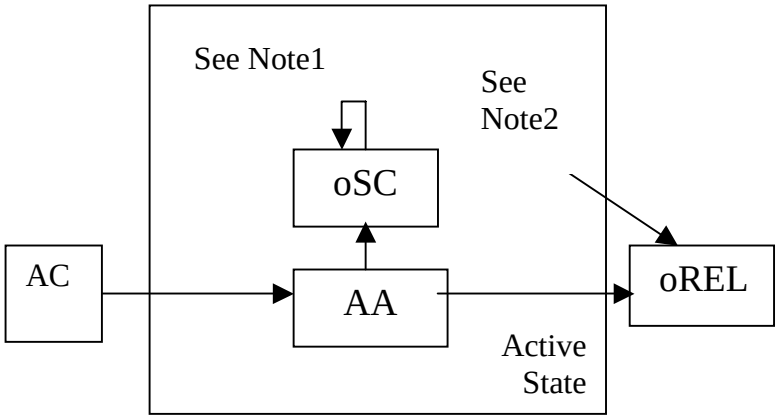


Figure : Application view on event reporting order Active State

Note 1: Only the detected service code or the range to which the service code belongs is disarmed as the service code is reported to the application
Note 2: The release event (oREL) can occur in any state resulting in a transition to Releasing state.

Abbreviations used for the events:

AC Address Collected; AA Address Analysed; oSC Originating Service Code; oREL Originating Release.

In this state the following functions are applicable:

- The detection of a Address_Analysed initial indication criterion.
- On receipt of the "Address_Analysed" indication the following functions are performed:

  i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_ADDRESS_ANALYSED then the event is intercepted and call leg processing is suspended.

  ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_ADDRESS_ANALYSED then the event is notified and call leg processing continues.

  iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_ADDRESS_ANALYSED then no monitoring is performed.

- Resumption of suspended call leg processing occurs on receipt of a continueProcessing() method.
- In this state the routing information is interpreted, the authority of the calling party to establish this connection is verified and the call leg connection is set up to the remote party.

- - In this state a connection to the call party is established.

- Detection of a "terminating release" indication (not visible to the application) from remote party caused by a network release event propagated from a terminating call leg causing the originating call leg STD to transit to Releasing state:

  Detection of a premature disconnect from the calling party.

- Receipt of a deassign() method.

- Receipt of a release() method.

- Detection of an "Answer" indication as a result of the remote party being connected (answered).

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Answer" initial indication criterion.

- On receipt of the "originating_service code" indication the following functions are performed:

  i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_ORIGINATING_SERVICE_CODE then the event is intercepted and call leg processing is suspended.

  ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_ORIGINATING_SERVICE_CODED then the event is notified and call leg processing continues..

  iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_ORIGINATING_SERVICE_CODE then no monitoring is performed.

- —Resumption of suspended call leg processing occurs on receipt of a continueProcessing() method.

Exit events:

- Detection of an "originating release" indication as a result of a disconnect from the calling party and and an "terminating release" indication as a result of a disconnect from called party.

- Receipt of a deassign() method.

- Receipt of a release() method from the application.

7.4.3.1.4 Releasing

Entry events:

- Detection of an "Originating_Release" or "Terminating Release" indication as a result of the network release initiated by calling party of called party..

- Reception of the release() method from ~~by~~ the application.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Originating_Release" initial indication criterion.

- A transition due to fault detection to this state is made when the Call leg object is in a state and no requests from the application have been received during a certain time period (timer expiry).

Functions:

In this state the connection to the call party is released as requested by the network or by the application and the reports are processed and sent to the application if requested .

When the Releasing state is entered the order of actions to be performed is as follows:
i) the network release event handling is performed.
ii) the possible call leg information requested with getInfoReq() and/ or superviseReq() is collected and send to the

application.
iii) the callLegEnded() method is sent to the application to inform that the call leg object is destroyed.

Where the entry to this state is caused by the application, for example because the application has requested the leg to be released or deassigned or a fault (e.g. timer expiry, no response from application) has been detected, then i) is not applicable. In the fault case for action ii) error report methods are sent to the application for any possible requested reports.

In this state the following functions are applicable:

- The detection of a "originating_release" initial indication criterion..
- On receipt of the "originating_release" indication the following functions are performed:
  - The network release event handling is performed as follows:
    i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_RELEASE then the event is intercepted and call leg processing is suspended.
    ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_RELEASE then the event is notified and call leg processing continues.
    iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_RELEASE then no monitoring is performed.
- Resumption of suspended call leg processing occurs on receipt of a continueProcessing() method.
- The possible call leg information requested with the getInfoReq() and/or superviseReq() is collected and sent to the application with respectively the getInfoRes() and/or superviseRes() methods.
- The callLegEnded() method is sent to the application after all information has been sent. In case that the application has not requested additional call leg related information the call leg object is destroyed immediately and additionally the application will also be informed that the connection has ended
- —In case of abnormal termination due to a fault and the application requested for call leg related information previously, the application will be informed that this information is not available and additionally the application is informed that the call leg object is destroyed (callLegEnded).
  Note: the call in the network may continue or be released, depending e.g. on the call state.
- In case the release() method is received -in Releasing state it will be discarded. The request from the application to release the leg is ignored in this case because release of the leg is already ongoing.

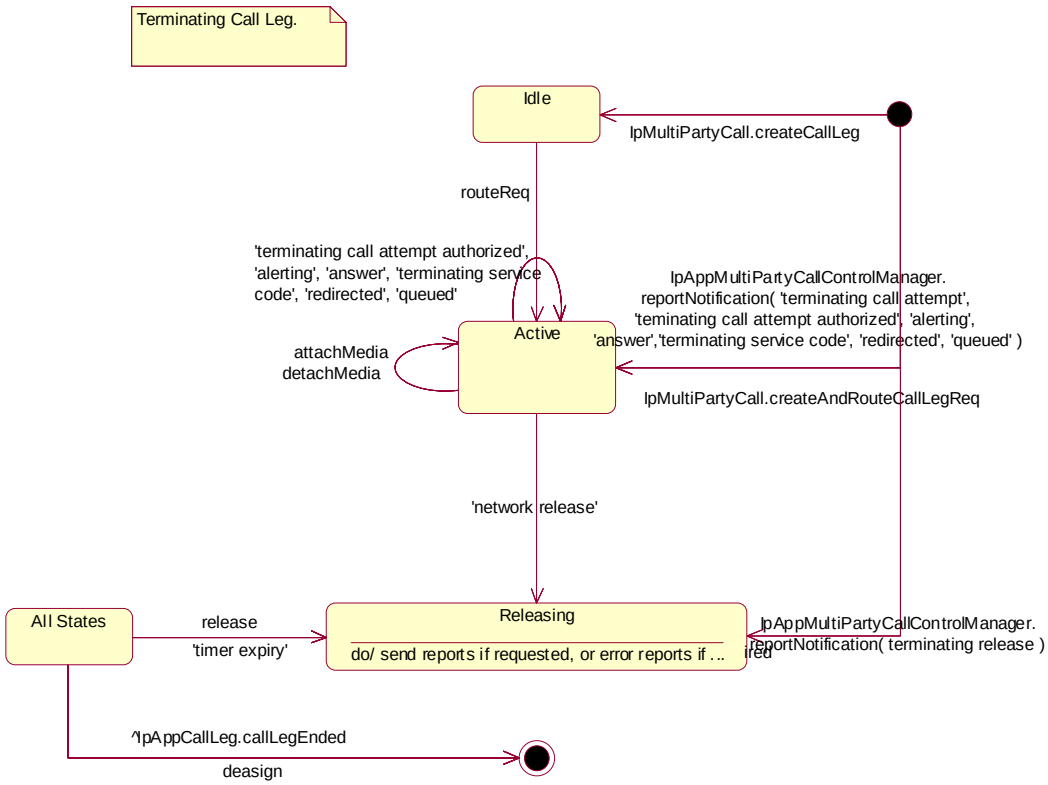
Exit events:

- In case that the application has not requested additional call leg related information the call leg object is destroyed immediately and additionally the application is informed that the call leg connection has ended, by sending the callLegEnded() method.
- Detection of the sending of the last call leg information to the application the Call Leg object is destroyed and additionally the application is informed that the call leg connection has ended, by sending the callLegEnded() method .

7.4.3.1.5 Overview of allowed- methods, Originating Call Leg STD

state	**methods allowed**
Initiating	attachMedia (as a request), detachMedia, (as a request) getCall , getLastRedirectedAddress, continueProcessing, release (call leg), deassign eventReportReq, getInfoReq, setChargePlan, setAdviceOfCharge, superviseReq
Analysing	attachMedia (as a request), detachMedia, (as a request) getCall , getLastRedirectedAddress, continueProcessing, release (call leg), deassign eventReportReq, getInfoReq, setChargePlan, setAdviceOfCharge, superviseReq
Active	attachMedia, detachMedia, getCall , getLastRedirectedAddress, continueProcessing, release , deassign eventReportReq, getInfoReq, setChargePlan, setAdviceOfCharge, superviseReq
Releasing	getCall , getLastRedirectedAddress, continueProcessing, release , deassign

7.4.3.2 Terminating Call Leg



Transitions/events not shown:
All states:
continueProcessing, getLastRedirectedAddress, getCall, sending getInfoRes, superviseRes: no state change,
All states except Releasing:
eventReportReq, setAdviceOfCharge, getInfoReq, superviseReq, setChargePlan.

When the application is notified in reportNotification of an call related network event associated with the Terminating Call Leg STD, then the Originating Call Leg STD is created and is initialized to be in the Active state.

Figure : Application view on the Terminating CallLeg object

7.4.3.2.1 Idle

Entry events:

- Receipt of a createCallLeg() method to start an application initiated call leg connection.

Functions:

In this state the call leg object is created and the interface connection is idled.
The application activity timer is being provided.

In this state the following functions are applicable:

- Invoking routeReq will result in a request to actually route the call leg object.

- Resumption of call leg processing occurs on receipt of a routeReq() method.

Exit events:

- Receipt of a routeReq() method from the application.

- Application activity timer expiry indicating that no requests from the application have been received during a certain period to continue processing.

- Receipt of a deassign() method.

- Receipt of a release() method.

- Detection of a network release event being an "originating release" indication as a result of a premature disconnect from the calling party.

7.4.3.2.2 Active

Entry events:

- Receipt of an routeReq will result in actually routing the call leg object.

- Receipt of a createAndRouteCallLeg() method to start an application initiated call leg connection.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Terminating_Call_Attempt" trigger criterion.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Terminating_Call_Attempt_Authorized" trigger criterion.

Functions:

In this state the routing information is interpreted, the authority of the called party to establish this connection is verified for the call leg connection. In this state a connection to the call party is established whereby events from the network may indicate to the application when the party is alerted (acknowledge connection setup) and when the party answer (confirmation of connection setup).
Furthermore, In this state terminating service code events can be received.

The figure below shows the order in which network events may be detected in the Active state and depending on the monitor mode be reported to the application.

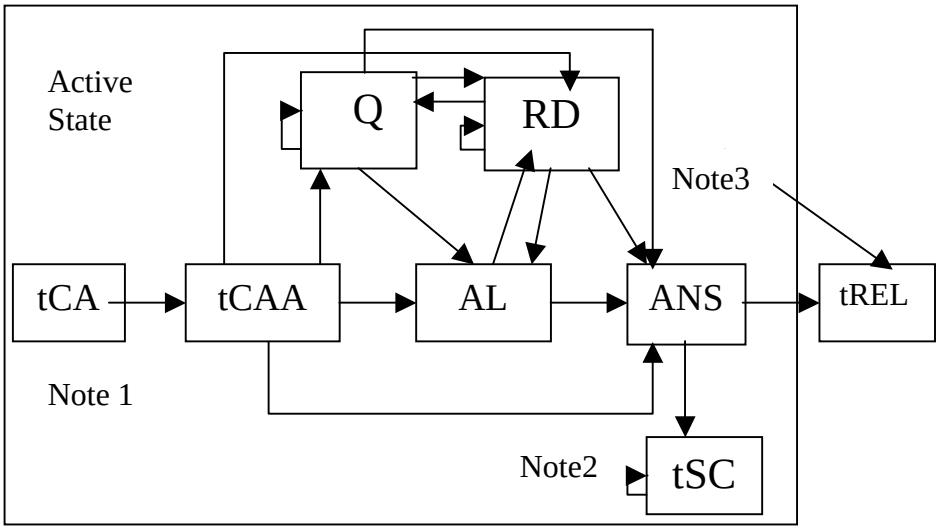


Figure : Application view on event reporting order in Active State

.
Note 1: Event tCA applicable as initial notification
Note 2: Only the detected service code or the range to which the service code belongs is disarmed as the service code is reported to the application
Note 3: The release event (tREL) can occur in any state resulting in a transition to Releasing state.

~~A~~Abbreviations used for the events:

tCA Terminating Call Attempt; tCAA Terminating Call Attempt Authorized; AL Alerting; ANS Answer; tREL Terminating Release; Q Queued; RD Redirected; tSC Terminating Service Code.

In this state the following functions are applicable:

- The detection of an "Terminating_Call_Attempt" initial notification criterion as a result that the call attempt.
- The detection of an "Terminating_Call_Attempt_Authorised" initial notification criterion as a result that the call attempt authorisation is successful.
- The report of the "Terminating_Call_Attempt_Authorised" event indication whereby the following functions are performed:
  - i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED then the event is intercepted and call leg processing is suspended.
  - ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED then the event is notified and call leg processing continues.
  - iv) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_CALL_TERMINATING_ATTEMPT_AUTHORISED then no monitoring is performed.
- Detection of an "Queued" indication as a result of the call to remote party being queued.
- On receipt of the "Queued" indication the following functions are performed:
  - i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_QUEUED then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_QUEUED then the event is notified and call leg processing continues.

iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_QUEUED then no monitoring is performed.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Alerting" trigger criterion.

- On receipt of the "Alerting" indication the following functions are performed:

i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_ALERTING then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_ALERTING then the event is notified and call leg processing continues.

iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_ALERTING then no monitoring is performed.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Answer" trigger criterion.

- Detection of an "Answer" indication as a result of the remote party being connected (answered).

- On receipt of the "Answer" indication the following functions are performed:

i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_ANSWER then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_ANSWER then the event is notified and call leg processing continues.

iv) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_ANSWER then no monitoring is performed.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "service_code" trigger criterion.

- The detection of a "service_code" trigger criterion suspends call leg processing.

- On receipt of the "service code" indication the following functions are performed:

i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_TERMINATING_SERVICE_CODE then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_TERMINATING_SERVICE_CODE then this is not a valid event (that event is not notified) and call leg processing continues.

iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_TERMINATING_SERVICE_CODE then no monitoring is performed.

- On receipt of the "redirected" indication the following functions are performed:

i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_REDIRECTED then the event is intercepted and call leg processing is suspended.

ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_REDIRECTED then the event is notified and call leg processing continues~~this is not a valid event (that event is not notified) and call leg processing continues~~.

iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_REDIRECTED then no monitoring is performed.

- Resumption of call leg processing occurs on receipt of a continueProcessing() method.

Exit events:

- Detection of a network release event being an "terminating release" indication as a result of the following events:

  i) Unable to select a route or indication from the remote party of the call leg connection cannot be presented (this is the network determined busy condition)

  ii) Occurrence of an authorisation failure when the authority to place the call leg connection was denied (e.g. business group restriction mismatch).

  iii) Detection of a route busy condition received from the remote call leg connection portion.

  iv) Detection of a no-answer condition received from the remote call leg connection portion.

  iv) Detection that the remote party was not reachable.

- Detection of a network release event being an "originating release" indication as a result of the following events:

  vi) Detection of a premature disconnect from the calling party.

- Receipt of a deassign() method.

- Receipt of a release() method from the application.

- Detection of a netwok release event being an "originating release" indication as a result of a disconnect from the calling party or a "terminating release" indication as a result of a disconnect from the called party.

7.4.3.2.3 Releasing

Entry events:

- Detection of a network release event being an "originating release" indication as a result of the network release initiated by calling party or a "terminating release" indication as a result of the network release initiated by called party..

- Sending of the release() method by the application.

- Sending of a reportNotification() method by the IPMultipartyCallControlManager for an "Terminating Release" trigger criterion.

- A transition due to fault detection to this state is made when the Call leg object awaits a request from the application and this is not received within a certain time period.

- Detection of a network event being a "terminating release" indication as a result of the following events:

  i) Unable to select a route or indication from the remote party of the call leg connection cannot be presented (this is the network determined busy condition)

  ii) Occurrence of an authorisation failure when the authority to place the call leg connection was denied (e.g. business group restriction mismatch).

  iii) Detection of a route busy condition received from the remote call leg connection portion.

  iv) Detection of a no-answer condition received from the remote call leg connection portion.

  v) Detection that the remote party was not reachable.

- Detection of a network release event being an "originating release" indication as a result of the following events:

  vi) Detection of a premature disconnect from the calling party.

Functions:

In this state the connection to the call party is released as requested by the network or by the application and the reports are processed and sent to the application if requested .

When the Releasing state is entered the order of actions to be performed is as follows:
i) the release event handling is performed.
ii) the possible call leg information requested with getInfoReq() and/ or superviseReq() is collected and send to the application.
iii) the callLegEnded() method is sent to the application to inform that the call leg object is destroyed.

Where the entry to this state is caused by the application, for example because the application has requested the leg to be released or deassigned or a fault (e.g. timer expiry, no response from application) has been detected, then i) is not applicable. In the fault case for action ii) error report methods are sent to the application for any possible requested reports.

In this state the following functions are applicable:

- The detection of a "Terminating Release" trigger criterion.

- On receipt of the network release event being a "Terminating Release" indication the following functions are performed:

  - The network release event handling is performed as follows:

    i) When the P_CALL_MONITOR_MODE_INTERRUPT is requested for the call leg event P_CALL_EVENT_TERMINATING_RELEASE then the event is intercepted and call leg processing is suspended.

    ii) When the P_CALL_MONITOR_MODE_NOTIFY is requested for the call leg event P_CALL_EVENT_TERMINATING_RELEASE then the event is notified and call leg processing continues.

    iii) When the P_CALL_MONITOR_MODE_DO_NOT_MONITOR is requested for the call leg event P_CALL_EVENT_TERMINATING_RELEASE then no monitoring is performed.

- Resumption of suspended call leg processing occurs on receipt of a continueProcessing() method.

- The possible call leg information requested with the getInfoReq() and/or superviseReq() is collected and sent to the application with respectively the getInfoRes() and/or superviseRes() methods.

- The callLegEnded() method is sent to the application after all information has been sent. In case that the application has not requested additional call leg related information the call leg object is destroyed immediately and additionally the application will also be informed that the connection has ended

- In case of abnormal termination due to a fault and the application requested for call leg related information previously, the application will be informed that this information is not available and additionally the application is informed that the call leg object is destroyed (callLegEnded).
  Note: the call in the network may continue or be released, depending e.g. on the call state.

- In case the release() method is received in Releasing state it will be discarded. The request from the application to release the leg is ignored in this case because release of the leg is already ongoing.

Exit events:

- In case that the application has not requested additional call leg related information the call leg object is destroyed immediately and additionally the application is informed that the call leg connection has ended, by sending the callLegEnded() method.

- Detection of the sending of the last call leg information to the application the Call Leg object is destroyed and additionally the application is informed that the call leg connection has ended, by sending the callLegEnded() method .

7.4.3.2.4 Overview of allowed methods and trigger events, Terminating Call Leg STD

state	**methods allowed**
Idle	routeReq, getCall , getLastRedirectedAddress, release, deassign eventReportReq, getInfoReq, setChargePlan, setAdviceOfCharge, superviseReq
Active	attachMedia detachMedia getCall , getLastRedirectedAddress, continueProcessing, release, deassign eventReportReq, getInfoReq, setChargePlan, setAdviceOfCharge, superviseReq
Releasing	- getCall , getLastRedirectedAddress, continueProcessing, release, deassign

~~7.4.3.1Idle State~~

~~In this state a new CallLeg object has been created and the application has not yet issued a routing request.~~

~~7.4.3.2Routing State~~

~~In this state a connection to the call party is being established.~~

~~7.4.3.3Connected State~~

~~In this state a connection to the call party is established.~~

~~In case the request for the connection was made by createAndRouteCallLeg on the Call object, the call party is also attached to the Call.~~

In case the request was made by route() the call party still needs to be attached to the Call.

7.4.3.4Failed or Disconnected State

In this state no connection to the call party could be established or the call party has disconnected.

The reason that no connection could be established can be that an invalid address was specified, the network aborted routing or the call party was busy.

7.4.3.5Incoming State

This state is only valid for an incoming Call Leg in case and there is no call established to another party.

7.4.3.6Progress State

In this sub-state the network has indicated there is progress in routing the CallLeg.

7.4.3.7Alerting State

In this sub-state the network has indicated there the terminal of the party is alerting.

7.4.3.8Redirected State

In this sub-state the network has indicated the call party has redirected calls to another address.

7.4.3.9Attached State

In this sub-state the media of the Call Leg object is attached to a Call object.

7.4.3.10Detached State

7.6 Multi-Party Call Control Data Definitions

The present document provides the MPCC~~GCC~~ data definitions necessary to support the~~-~~ API specification.

The general format of a data definition specification is described below.

- Data Type

This shows the name of the data type.

- Description

This describes the data type.

- Tabular Specification

This specifies the data types and values of the data type.

- Example

If relevant, an example is shown to illustrate the data type.

7.6.1 Event Notification Data Definitions

No specific event notification data defined.

7.6.2 Multi-Party Call Control Data Definitions

…

TpCallEventType

Defines a specific call event report type.

Name	Value	Description
P_CALL_EVENT_UNDEFINED	0	Undefined
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	1	A originating ~~C~~call attempt takes place (e.g. Off-hook event).
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORIZED	2	A originating call attempt is authorized
P_CALL_EVENT_ADDRESS_COLLECTED	~~2~~3	The destination address has been collected.
P_CALL_EVENT_ADDRESS_ANALYSED	~~4~~3	The destination address has been analysed.
P_CALL_EVENT_ORIGINATING_SERVICE_CODE	5	Mid-call originating service code received.
P_CALL_EVENT_ORIGINATING_RELEASE	6	A originating call/call leg is released
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	7	A terminating call attempt takes place
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED	8	A terminating call is authorized
P_CALL_EVENT_ALERTING	9~~5~~	Call is alerting at the call party.
P_CALL_EVENT_ANSWER	10~~6~~	Call answered at address.
P_CALL_EVENT_TERMINATING_RELEASE	11~~7~~	A terminating c~~C~~all/call leg is~~has been~~ released or the call could not be routed.
P_CALL_EVENT_REDIRECTED	12~~8~~	Call redirected to new address: an indication from the network that the call has been redirected to a new address (no events are disarmed as a result of this).
P_CALL_EVENT_TERMINATING_SERVICE_CODE	13~~9~~	Mid-~~call~~ call terminating service code received.
P_CALL_EVENT_QUEUED	~~14~~10	The Call Event has been queued. (no events are disarmed as a result of this)

EVENT HANDLING RULES:

The following general event handling rules apply to dynamically armed events:

- If an armed event is met, then it is disarmed, unless explicit stated that it will not to be disarmed.
- If an event is met that causes the release of the related leg, then all events related to that leg are disarmed .
- When an event is met on a call leg irrespective of the event monitor mode, then only events belonging to that call leg may become disarmed (see table below) .
- If a call is released, then all events related to that call are disarmed.

Note: Event disarmed means monitor mode is set to DO_NOT_MONITOR. and
event armed means monitor mode is set to INTERRUPT or NOTIFY..

The table below defines the disarming rules for dynamic events. In case such an event occurs on a call leg the table shows which events are disarmed (are not monitored anymore) on that call leg and should be re-armed by eventReportReq() in case the application is still interested in these events.

Event Occurred	Events Disarmed
P_CALL_EVENT_UNDEFINED	Not Applicable
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	Not applicable, can only be armed as trigger
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORIZED	P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORIZED
P_CALL_EVENT_ADDRESS_COLLECTED	P_CALL_EVENT_ADDRESS_COLLECTED
P_CALL_EVENT_ADDRESS_ANALYSED	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED
~~P_CALL_EVENT_PROGRESS~~	~~P_CALL_EVENT_ADDRESS_COLLECTED~~ ~~P_CALL_EVENT_ADDRESS_ANALYSED~~ ~~P_CALL_EVENT_PROGRESS~~
P_CALL_EVENT_ALERTING	~~P_CALL_EVENT_ADDRESS_COLLECTED~~ ~~P_CALL_EVENT_ADDRESS_ANALYSED~~ ~~P_CALL_EVENT_PROGRESS~~ P_CALL_EVENT_ALERTING P_CALL_EVENT_TERMINATING_RELEASE with criteria: P_USER_NOT_AVAILABLE P_BUSY P_NOT_REACHABLE P_ROUTING_FAILURE P_CALL_RESTRICTED P_UNAVAILABLE_RESOURCES
P_CALL_EVENT_ANSWER	~~P_CALL_EVENT_ADDRESS_COLLECTED~~ ~~P_CALL_EVENT_ADDRESS_ANALYSED~~ ~~P_CALL_EVENT_PROGRESS~~ ~~P_CALL_EVENT_ANSWER~~ P_CALL_EVENT_ALERTING P_CALL_EVENT_TERMINATING_RELEASE with criteria: P_USER_NOT_AVAILABLE P_BUSY P_NOT_REACHABLE P_ROUTING_FAILURE P_CALL_RESTRICTED P_UNAVAILABLE_RESOURCES P_NO_ANSWER ~~P_PREMATURE_DISCONNECT~~ ~~P_CALL_EVENT_ANSWER~~
P_CALL_EVENT_ORIGINATING_RELEASE	All pending network events for the call leg are disarmed
P_CALL_EVENT_TERMINATING_RELEASE	All pending network events for the call leg are disarmed
P_CALL_EVENT_ORIGINATING_SERVICE_CODE ~~P_CALL_EVENT_REDIRECTED~~	P_CALL_EVENT_ORIGINATING_SERVICE_CODE_*) see NOTE1 ~~P_CALL_EVENT_REDIRECTED~~
P_CALL_EVENT_TERMINATING_SERVICE_CODE	P_CALL_EVENT_TERMINATING_SERVICE_CODE_*) see NOTE1

NOTE 1: Only the detected service code or the range to which the service code belongs is disarmed.

TpAdditionalCallEventCriteria

Defines the Tagged Choice of Data Elements that specify specific criteria.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORIZED	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpInt32	MinAddressLength
P_CALL_EVENT_ADDRESS_ANALYSED	NULL	Undefined
P_CALL_EVENT_ORIGINATING_SERVICE_CODE ~~PROGRESS~~	TpCallServiceCode ~~NULL~~	ServiceCode ~~Undefined~~
P_CALL_EVENT_ORIGINATING_RELEASE	TpCallReleaseCauseSet	ReleaseCauseSet
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORIZED	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_TERMINATING_RELEASE	TpCallReleaseCauseSet	ReleaseCauseSet
P_CALL_EVENT_REDIRECTED	NULL	Undefined
P_CALL_EVENT_TERMINATING_SERVICE_CODE	TpCallServiceCode	ServiceCode
P_CALL_EVENT_QUEUED	NULL	Undefined

TpCallAdditionalEventInfo

Defines the Tagged Choice of Data Elements that specify additional call event information for certain types of events.

	Tag Element Type	
	TpCallEventType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_EVENT_UNDEFINED	NULL	Undefined
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORIZED	NULL	Undefined
P_CALL_EVENT_ADDRESS_COLLECTED	TpAddress	CollectedAddress
P_CALL_EVENT_ADDRESS_ANALYSED	TpAddress	CalledAddress
P_CALL_EVENT_ORIGINATING_SERVICE_CODE~~PROGRESS~~	NULL	Undefined
P_CALL_EVENT_ORIGINATING_RELEASE	TpCallReleaseCause	ReleaseCause
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	NULL	Undefined
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORIZED	NULL	Undefined
P_CALL_EVENT_QUEUED	NULL	Undefined
P_CALL_EVENT_ALERTING	NULL	Undefined
P_CALL_EVENT_ANSWER	NULL	Undefined
P_CALL_EVENT_TERMINATING_RELEASE	TpCallReleaseCause	ReleaseCause
P_CALL_EVENT_REDIRECTED	TpAddress	ForwardAddress
P_CALL_EVENT_TERMINATING_SERVICE_CODE	TpCallServiceCode	ServiceCode

TpCallNotificationScope

Defines a the sequence of Data elements that specify the scope of a notification request.

Of the addresses only the Plan and the AddrString are used for the purpose of matching the notifications against the criteria.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddressRange	Defines the destination address or address range for which the notification is requested.
OriginatingAddress	TpAddressRange	Defines the origination address or address range for which the notification is requested.
~~NotificationCallType~~	~~TpNotificationCallType~~	~~Defines wheter the notification is requested for a originating or terminating call.~~

~~TpNotificationCallType~~

~~Defines the type of call for which the notification is requested or reported.~~

~~Name~~	~~Value~~	~~Description~~
~~P_ORIGINATING~~	~~1~~	~~Indicates that the notification is related to the originating user in the call.~~
~~P_TERMINATING~~	~~2~~	~~Indicates that the notification is related to the terminating user in the call.~~

TpCallNotificationReportScope

Defines the Sequence of Data Elements that specify the scope for which a notification report was sent.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddress	Contains the destination address of the call.
OriginatingAddress	TpAddress	Contains the origination address of the call
~~NotificationCallType~~	~~TpNotificationCallType~~	~~Indicates if the notification was reported for an originating or terminating call.~~

…

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010631**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **008** ⌘ ev **-** ⌘ Current version: **4.0.0.** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to SetChargePlan() Addition of PartyToCharge parmeter
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	PartyToCharge parameter must be added to TpCallChargePlan to the SetChargePlan() methods for IpMultiPartyCall and IpCallLeg
Summary of change:	⌘	Parameter PartyToCharge must be added to TpCallChargePlan to the SetChargePlan() methods for IpMultiPartyCall and IpCallLeg
Consequences if not approved:	⌘	Missing parameter.

Clauses affected:	⌘	Section 6.6.2 for SetChargePlan() methods in Sections 7.3.3.and 7.3.5
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

* **Contact:** Frans Haerens ☎+32-3-240.90.34 / ✉ frans.haerens@alcatel.be
Freddy Ghys ☎+32-3-240.95.37 / ✉ freddy.ghys@alcatel.be

1 Introduction

It is proposed that the TpCallPartyToCharge is added to both the multiparty call and call leg setChargePlan methods in the Draft ETSI 201 915-4. It is proposed that the TpCallPartyToCharge parameter as a minimum can identify the parties to which the application charging applies e.g. a party belonging to the call, a party related to the call leg, a third party, and a service provider. It is therefore proposed to have the following tag element values : P_CALL_PARTY_ORIGINATING, P_CALL_PARTY_DESTINATION and P_CALL_PARTY_SPECIAL . Since in some of the case P_CALL_PARTY_SPECIAL the charge determination application instance has to provide the address the choice element type TpAddress is added for those instances.

2 Proposed corrections

TpCallChargePlan

Defines the Sequence of Data Elements that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
ChargeOrderType	TpCallChargeOrderCategory	Charge order
~~ChargePerTime~~	~~TpChargePerTime~~	~~Charge per time.~~ ~~Only applicable when time based charging is selected.~~
TransparentCharge	TpOctetSet	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
ChargePlan	TpInt32	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when ~~transparent~~ predefined charging is selected.
~~Currency~~	~~TpString~~	~~Currency unit according to ISO-4217:1995~~
AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
PartyToCharge	TpCallPartyToCharge	Identifies the entity or party to be charged for the call or call leg.

TpCallPartyToCharge

Defines the Tagged Choice of Data Elements that identifies the entity or party to be charged.

	Tag Element Type		

	TpCallPartyToChargeType		

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_PARTY_ORIGINATING, ,	NULL	Undefined
P_CALL_PARTY_DESTINATION,	NULLf	Undefined
P_CALL_PARTY_SPECIAL	TpAddress	CallPartySpecial

TpCallPartyToChargeType

Defines the type of call party to charge

Name	Value	Description
P_CALL_PARTY_ORIGINATING, ,	0	Calling party, i.e. party that initiated the call. For application initiated calls this indicates the first party of the call
P_CALL_PARTY_DESTINATION,	1	Called party
P_CALL_PARTY_SPECIAL	2	An address identifying e.g. a third party, a service provider

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010635**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **009** ⌘ ev **-** ⌘ Current version: **4.0.0.** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to SetChargePlan()
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	We noted some technical flaws and corrections on the SetChargePlan() methods for IpMultiPartyCall and IpCallLeg
Summary of change:	⌘	Parameters TpCallChargePlan and TpCallChrgeOrderCategory needs corrections.
Consequences if not approved:	⌘	Correction of technical flaws needs resolution.

Clauses affected:	⌘	Section 6.6.2 for SetChargePlan() methods in Sections 7.3.3.and 7.3.5
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1 Introduction

It is proposed that the following technical flaws and corrections on the SetChargePlan() method parameters are introduced in the Draft ETSI 201 915-4.

2 Proposed corrections

The following parameters in the SetChargePlan() methods of the Interface Class IpMultiPartyCall (Section 7.3.3) and Interface Class IpCallLeg (Section 7.3.5) need to be corrected into section 6.6.2.

TpCallChargePlan

Defines the Sequence of Data Elements that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
ChargeOrderType	TpCallChargeOrderCategory	Charge order
~~ChargePerTime~~	~~TpChargePerTime~~	~~Charge per time.~~ ~~Only applicable when time based charging is selected.~~
TransparentCharge	TpOctetSet	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
ChargePlan	TpInt32	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when ~~transparent~~ predefined charging is selected.
~~Currency~~	~~TpString~~	~~Currency unit according to ISO 4217:1995~~
AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.

TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
~~P_CALL_CHARGE_PER_TIME~~	~~0~~	~~Charge per time~~
P_CALL_CHARGE_TRANSPARENT	0~~1~~	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records
P_CALL_CHARGE_PREDEFINED_SET	1~~2~~	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user).

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010640**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **010** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Remove distinction between final- and intermediate-report
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	Problem is that in the case of non follow-on calls, an intermediate or final report is the same. In case of a follow-on call the final report contains data that is already available to the application (by means of the intermediate reports).
Summary of change:	⌘	The distinction between a final- and intermediate report should be removed. In case of follow-on calls each time the called party disconnects or when the call is ended a report will be generated and sent to the application. Also we would like to make the description of the call times more clear.
Consequences if not approved:	⌘	Insufficient reporting function

Clauses affected:	⌘	Chapter 6.3.3 and 6.6.2 for GCC. And 7.3.3 for MPCC chapter. Chapter 8 is common.
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

6 Generic Call Control Service

6.3.3 Interface Class IpCall

Method

`getCallInfoReq()`

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method shall be invoked before the call is routed to a target address. ~~Two types of reports can be requested; a final report or intermediate reports.~~

~~A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.~~

A~~Intermediate~~ report~~s~~ is~~are~~ received when the destination leg or party terminates or when the call ends. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call.In case the originating party is still available the application can still initiate a follow-on call using routeReq.

Parameters

callSessionID : in TpSessionID
Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType
Specifies the call information that is requested.

Raises

TpGCCSException,TpGeneralException

6.6.2 Generic Call Control Data Definitions

`TpCallInfoReport`

Defines the `Sequence of Data Elements` that specify the call information requested. Information that was not requested is invalid.

Sequence Element Name	Sequence Element Type	Description
`CallInfoType`	TpCallInfoType	The type of call report.
`CallInitiationStartTime`	`TpDateAndTime`	The time and date when the call, or follow-on call, was started as a result of a routeReq.
`CallConnectedToResourceTime`	`TpDateAndTime`	The date and time when the call was connected to the resource. This data element is only valid when information on user interaction is reported.
`CallConnectedToDestinationTime`	`TpDateAndTime`	The date and time when the call 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. This data element is invalid when information on user interaction is reported~~ with an intermedia report~~.
`CallEndTime`	`TpDateAndTime`	The date and time when the call or follow-on call or user interaction was terminated.
`Cause`	`TpCallReleaseCause`	The cause of the termination.

A callInfoReport will be generated at the end of user interaction and at the end of the connection with the associated address. This means that either the destination related information is present or the resource related information, but not both.

7 Multi-Party Call Control Service

7.3.3 Interface Class IpMultiPartyCall

Method

getInfoReq()

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method shall be invoked before the call is routed to a target address. ~~Two types of reports can be requested; a final report or intermediate reports.~~

~~A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.~~

A report~~Intermediate reports are~~is received when the destination leg or party terminates or when the call ends. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. In case the originating party is still available the application can still initiate a follow-on call using routeReq.

Parameters

callSessionID : in TpSessionID
Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType
Specifies the call information that is requested.

Raises

TpGCCSException,TpGeneralException

8 Common Call Control Data Types

TpCallInfoType

Defines the type of call information requested and reported. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_INFO_UNDEFINED	00h	Undefined
P_CALL_INFO_TIMES	01h	Relevant call times
P_CALL_INFO_RELEASE_CAUSE	02h	Call release cause
~~P_CALL_INFO_INTERMEDIATE~~	~~04h~~	~~Send only intermediate reports. When this is not specified the information report will only be sent when the call has ended. When intermediate reports are requested a report will be generated between follow-on calls, i.e. when a party leaves the call.~~

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010648**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **011** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Inclusion of TpMediaType
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	One of the Service properties for call control is the media used by the service. In that section a reference is made to the data-type TpMediaType. However, this data-type is at the moment assigned to the multi-media call control and thus not present in the TS 29.198-4.
Summary of change:	⌘	Introduction of the data-type TpMediaType in 29.198-4
Consequences if not approved:	⌘	incomplete specification.

Clauses affected:	⌘	5
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications ⌘
Other comments:	⌘	

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

5 Common Call Control Data Types

TpCallAlertingMechanism

This data type is identical to a TpInt32, and defines the mechanism that will be used to alert a call party. The values of this data type are operator specific.

TpCallBearerService

This data type defines the type of call application-related specific information (Q.931: Information Transfer Capability, and 3G TS 22.002).

Name	Value	Description
P_CALL_BEARER_SERVICE_UNKNOWN	0	Bearer capability information unknown at this time
P_CALL_BEARER_SERVICE_SPEECH	1	Speech
P_CALL_BEARER_SERVICE_DIGITALUNRESTRICTED	2	Unrestricted digital information
P_CALL_BEARER_SERVICE_DIGITALRESTRICTED	3	Restricted digital information
P_CALL_BEARER_SERVICE_AUDIO	4	3.1 kHz audio
P_CALL_BEARER_SERVICE_ DIGITALUNRESTRICTEDTONES	5	Unrestricted digital information with tomes/announcements
P_CALL_BEARER_SERVICE_VIDEO	6	Video

TpCallChargePlan

Defines the Sequence of Data Elements that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
ChargeOrderType	TpCallChargeOrderCategory	Charge order
ChargePerTime	TpChargePerTime	Charge per time. Only applicable when time based charging is selected.
TransparentCharge	TpOctetSet	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
ChargePlan	TpInt32	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when transparent charging is selected.
Currency	TpString	Currency unit according to ISO-4217:1995

AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.

Valid Currencies are:

ADP, AED, AFA, ALL, AMD, ANG, AON, AOR, ARS, ATS, AUD, AWG, AZM, BAM, BBD, BDT, BEF, BGL, BGN, BHD, BIF, BMD, BND, BOB, BOV, BRL, BSD, BTN, BWP, BYB, BZD, CAD, CDF, CHF, CLF, CLP, CNY, COP, CRC, CUP, CVE, CYP, CZK, DEM, DJF, DKK, DOP, DZD, ECS, ECV, EEK, EGP, ERN, ESP, ETB, EUR, FIM, FJD, FKP, FRF, GBP, GEL, GHC, GIP, GMD, GNF, GRD, GTQ, GWP, GYD, HKD, HNL, HRK, HTG, HUF, IDR, IEP, ILS, INR, IQD, IRR, ISK, ITL, JMD, JOD, JPY, KES, KGS, KHR, KMF, KPW, KRW, KWD, KYD, KZT, LAK, LBP, LKR, LRD, LSL, LTL, LUF, LVL, LYD, MAD, MDL, MGF, MKD, MMK, MNT, MOP, MRO, MTL, MUR, MVR, MWK, MXN, MXV, MYR, MZM, NAD, NGN, NIO, NLG, NOK, NPR, NZD, OMR, PAB, PEN, PGK, PHP, PKR, PLN, PTE, PYG, QAR, ROL, RUB, RUR, RWF, SAR, SBD, SCR, SDD, SEK, SGD, SHP, SIT, SKK, SLL, SOS, SRG, STD, SVC, SYP, SZL, THB, TJR, TMM, TND, TOP, TPE, TRL, TTD, TWD, TZS, UAH, UGX, USD, USN, USS, UYU, UZS, VEB, VND, VUV, WST, XAF, XAG, XAU, XBA, XBB, XBC, XBD, XCD, XDR, XFO, XFU, XOF, XPD, XPF, XPT, XTS, XXX, YER, YUM, ZAL, ZAR, ZMK, ZRN, ZWD.

XXX is used for transactions where no currency is involved.

TpCallChargeOrder

Defines the Tagged Choice of Data Elements that specify the charge plan for the call.

	Tag Element Type	
	TpCallChargeOrderCategory	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_CHARGE_PER_TIME	TpChargePerTime	ChargePerTime
P_CALL_CHARGE_TRANSPARENT	TpOctetSet	TransparentCharge
P_CALL_CHARGE_PREDEFINED_SET	TpInt32	ChargePlan

TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
P_CALL_CHARGE_PER_TIME	0	Charge per time
P_CALL_CHARGE_TRANSPARENT	1	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records
P_CALL_CHARGE_PREDEFINED_SET	2	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold

		card user).

TpCallAdditionalChargePlanInfo

Defines the Tagged Choice of Data Elements that specify the charge plan for the call.

	Tag Element Type	
	TpCallChargeOrderCategory	

Tag Element Value	Choice Element Type	Choice Element Name	Description
P_CALL_CHARGE_PER_TIME	TpOctetSet	TimeAdditionalInfo	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
P_CALL_CHARGE_TRANSPARENT	NULL	Undefined	
P_CALL_CHARGE_PREDEFINED_SET	TpOctetSet	SetAdditionalInfo	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.

TpCallEndedReport

Defines the Sequence of Data Elements that specify the reason for the call ending.

Sequence Element Name	Sequence Element Type	
CallLegSessionID	TpSessionID	The leg that initiated the release of the call. If the call release was not initiated by the leg, then this value is set to –1.
Cause	TpCallReleaseCause	The cause of the call ending.

TpCallError

Defines the Sequence of Data Elements that specify the additional information relating to acall error.

Sequence Element Name	Sequence Element Type
ErrorTime	TpDateAndTime
ErrorType	TpCallErrorType
AdditionalErrorInfo	TpCallAdditionalErrorInfo

TpCallAdditionalErrorInfo

Defines the Tagged Choice of Data Elements that specify additional call error and call error specific information. This is also used to specify call leg errors and information errors.

	Tag Element Type	
	TpCallErrorType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_ERROR_UNDEFINED	NULL	Undefined
P_CALL_ERROR_INVALID_ADDRESS	TpAddressError	CallErrorInvalidAddress
P_CALL_ERROR_INVALID_STATE	NULL	Undefined

TpCallErrorType

Defines a specific call error.

Name	Value	Description
P_CALL_ERROR_UNDEFINED	0	Undefined; the method failed or was refused, but no specific reason can be given.
P_CALL_ERROR_INVALID_ADDRESS	1	The operation failed because an invalid address was given
P_CALL_ERROR_INVALID_STATE	2	The call was not in a valid state for the requested operation

TpCallFault

Defines the cause of the call fault detected.

Name	Value	Description
P_CALL_FAULT_UNDEFINED	0	Undefined
P_CALL_TIMEOUT_ON_RELEASE	1	This fault occurs when the final report has been sent to the application, but the application did not explicitly release or deassign the call object, within a specified time. The timer value is operator specific.
P_CALL_TIMEOUT_ON_INTERRUPT	2	This fault occurs when the application did not instruct the gateway how to handle the call within a specified time, after the gateway reported an event that was requested by the application in interrupt mode. The timer value is operator specific.

TpCallInfoReport

Defines the Sequence of Data Elements that specify the call information requested. Information that was not requested is invalid.

Sequence Element Name	Sequence Element Type	Description
CallInfoType	TpCallInfoType	The type of call report.
CallInitiationStartTime	TpDateAndTime	The time and date when the call, or follow-on call, was started.
CallConnectedToResourceTime	TpDateAndTime	The date and time when the call was connected to the resource. This data element is only valid when information on user interaction is reported.
CallConnectedToDestinationTime	TpDateAndTime	The date and time when the call 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. This data element is invalid when information on user interaction is reported with an intermediate report.
CallEndTime	TpDateAndTime	The date and time when the call or follow-on call or user interaction was terminated.
Cause	TpCallReleaseCause	The cause of the termination.

A callInfoReport will be generated at the end of user interaction and at the end of the connection with the associated address. This means that either the destination related information is present or the resource related information, but not both.

TpCallInfoType

Defines the type of call information requested and reported. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_INFO_UNDEFINED	00h	Undefined
P_CALL_INFO_TIMES	01h	Relevant call times
P_CALL_INFO_RELEASE_CAUSE	02h	Call release cause
P_CALL_INFO_INTERMEDIATE	04h	Send only intermediate reports. When this is not specified the information report will only be sent when the call has ended. When intermediate reports are requested a report will be generated between follow-on calls, i.e., when a party leaves the call.

TpCallLoadControlMechanism

Defines the Tagged Choice of Data Elements that specify the applied mechanism and associated parameters.

	Tag Element Type	
	TpCallLoadControlMechanismType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_LOAD_CONTROL_PER_INTERVAL	TpCallLoadControlIntervalRate	CallLoadControlPerInterval

TpCallLoadControlIntervalRate

Defines the call admission rate of the call load control mechanism used. This data type indicates the interval (in milliseconds) between calls that are admitted.

Name	Value	Description
P_CALL_LOAD_CONTROL_ADMIT_NO_CALLS	0	Infinite interval (do not admit any calls)
	1 - 60000	Duration in milliseconds

TpCallLoadControlMechanismType

Defines the type of call load control mechanism to use.

Name	Value	Description
P_CALL_LOAD_CONTROL_PER_INTERVAL	1	admit one call per interval

TpCallMonitorMode

Defines the mode that the call will monitor for events, or the mode that the call is in following a detected event.

Name	Value	Description
P_CALL_MONITOR_MODE_INTERRUPT	0	The call event is intercepted by the call control service and call processing is interrupted. The application is notified of the event and call processing resumes following an appropriate API call or network event (such as a call release)
P_CALL_MONITOR_MODE_NOTIFY	1	The call event is detected by the call control service but not intercepted. The application is notified of the event and call processing

		continues
P_CALL_MONITOR_MODE_DO_NOT_MONITOR	2	Do not monitor for the event

TpCallNetworkAccessType

This data defines the bearer capabilities associated with the call. (3G TS 24.002) This information is network operator specific and may not always be available because there is no standard protocol to retrieve the information.

Name	Value	Description
P_CALL_NETWORK_ACCESS_TYPE_UNKNOWN	0	Network type information unknown at this time
P_CALL_NETWORK_ACCESS_TYPE_POT	1	POTS
P_CALL_NETWORK_ACCESS_TYPE_ISDN	2	ISDN
P_CALL_NETWORK_ACCESS_TYPE_DIALUPINTERNET	3	Dial-up Internet
P_CALL_NETWORK_ACCESS_TYPE_XDSL	4	xDLS
P_CALL_NETWORK_ACCESS_TYPE_WIRELESS	5	Wireless

TpCallPartyCategory

This data type defines the category of a calling party. (Q.763: Calling Party Category / Called Party Category)

Name	Value	Description
P_CALL_PARTY_CATEGORY_UNKNOWN	0	calling party's category unknown at this time
P_CALL_PARTY_CATEGORY_OPERATOR_F	1	operator, language French
P_CALL_PARTY_CATEGORY_OPERATOR_E	2	operator, language English
P_CALL_PARTY_CATEGORY_OPERATOR_G	3	operator, language German
P_CALL_PARTY_CATEGORY_OPERATOR_R	4	operator, language Russian
P_CALL_PARTY_CATEGORY_OPERATOR_S	5	operator, language Spanish
P_CALL_PARTY_CATEGORY_ORDINARY_SUB	6	ordinary calling subscriber
P_CALL_PARTY_CATEGORY_PRIORITY_SUB	7	calling subscriber with priority
P_CALL_PARTY_CATEGORY_DATA_CALL	8	data call (voice band data)
P_CALL_PARTY_CATEGORY_TEST_CALL	9	test call
P_CALL_PARTY_CATEGORY_PAYPHONE	10	payphone

TpCallServiceCode

Defines the Sequence of Data Elements that specify the service code and type of service code received during a call. The service code type defines how the value string should be interpreted.

Sequence Element Name	Sequence Element Type
CallServiceCodeType	TpCallServiceCodeType
ServiceCodeValue	TpString

TpCallServiceCodeType

Defines the different types of service codes that can be received during the call.

Name	Value	Description
P_CALL_SERVICE_CODE_UNDEFINED	0	The type of service code is unknown. The corresponding string is operator specific.
P_CALL_SERVICE_CODE_DIGITS	1	The user entered a digit sequence during the call. The corresponding string is an ascii representation of the received digits.
P_CALL_SERVICE_CODE_FACILITY	2	A facility information element is received. The corresponding string contains the facility information element as defined in ITU Q.932
P_CALL_SERVICE_CODE_U2U	3	A user-to-user message was received. The associated string contains the content of the user-to-user information element.
P_CALL_SERVICE_CODE_HOOKFLASH	4	The user performed a hookflash, optionally followed by some digits. The corresponding string is an ascii representation of the entered digits.
P_CALL_SERVICE_CODE_RECALL	5	The user pressed the register recall button, optionally followed by some digits. The corresponding string is an ascii representation of the entered digits.

TpCallSuperviseReport

Defines the responses from the call control service for calls that are supervised. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_SUPERVISE_TIMEOUT	01h	The call supervision timer has expired
P_CALL_SUPERVISE_CALL_ENDED	02h	The call has ended, either due to timer expiry or call party release. In case the called party disconnects but a follow-on call can still be made also this indication is used.
P_CALL_SUPERVISE_TONE_APPLIED	04h	A warning tone has been applied. This is only sent in combination with P_CALL_SUPERVISE_TIMEOUT
P_CALL_SUPERVISE_UI_FINISHED	0	The user interaction has finished.

TpCallSuperviseTreatment

Defines the treatment of the call by the call control service when the call supervision timer expires. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_CALL_SUPERVISE_RELEASE	01h	Release the call when the call supervision timer expires
P_CALL_SUPERVISE_RESPOND	02h	Notify the application when the call supervision timer expires
P_CALL_SUPERVISE_APPLY_TONE	04h	Send a warning tone to the originating party when the call supervision timer expires. If call release is requested, then the call will be released following the tone after an administered time period

TpCallTeleService

This data type defines the tele-service associated with the call. (Q.763: User Teleservice Information, Q.931: High Layer Compatitibility Information, and 3G TS 22.003)

Name	Value	Description
P_CALL_TELE_SERVICE_UNKNOWN	0	Teleservice information unknown at this time
P_CALL_TELE_SERVICE_TELEPHONY	1	Telephony
P_CALL_TELE_SERVICE_FAX_2_3	2	Facsimile Group 2/3
P_CALL_TELE_SERVICE_FAX_4_I	3	Facsimile Group 4, Class I
P_CALL_TELE_SERVICE_FAX_4_II_III	4	Facsimile Group 4, Classes II and III
P_CALL_TELE_SERVICE_VIDEOTEX_SYN	5	Syntax based Videotex
P_CALL_TELE_SERVICE_VIDEOTEX_INT	6	International Videotex interworking via gateways or interworking units
P_CALL_TELE_SERVICE_TELEX	7	Telex service
P_CALL_TELE_SERVICE_MHS	8	Message Handling Systems
P_CALL_TELE_SERVICE_OSI	9	OSI application
P_CALL_TELE_SERVICE_FTAM	10	FTAM application
P_CALL_TELE_SERVICE_VIDEO	11	Videotelephony
P_CALL_TELE_SERVICE_VIDEO_CONF	12	Videoconferencing
P_CALL_TELE_SERVICE_AUDIOGRAPH_CONF	13	Audiographic conferencing
P_CALL_TELE_SERVICE_MULTIMEDIA	14	Multimedia services
P_CALL_TELE_SERVICE_CS_INI_H221	15	Capability set of initial channel of H.221
P_CALL_TELE_SERVICE_CS_SUB_H221	16	Capability set of subsequent channel of H.221
P_CALL_TELE_SERVICE_CS_INI_CALL	17	Capability set of initial channel associated with an active 3.1 kHz audio or speech call.
P_CALL_TELE_SERVICE_DATATRAFFIC	18	Data traffic.
P_CALL_TELE_SERVICE_EMERGENCY_CALLS	1	Emergency Calls
P_CALL_TELE_SERVICE_SMS_MT_PP	2	Short message MT/PP
P_CALL_TELE_SERVICE_SMS_MO_PP	2	Short message MO/PP
P_CALL_TELE_SERVICE_CELL_BROADCAST	2	Cell Broadcast Service
P_CALL_TELE_SERVICE_ALT_SPEECH_FAX_3	2	Alternate speech and facsimile group 3
P_CALL_TELE_SERVICE_AUTOMATIC_FAX_3	2	Automatic Facsimile group 3
P_CALL_TELE_SERVICE_VOICE_GROUP_CALL	2	Voice Group Call Service
P_CALL_TELE_SERVICE_VOICE_BROADCAST	2	Voice Broadcast Service

TpCallTreatment

Defines the Sequence of Data Elements that specify the the treatment for calls that will be handled only by the network (for example, call which are not admitted by the call load control mechanism).

Sequence Element Name	Sequence Element Type
ReleaseCause	TpCallReleaseCause
AdditionalTreatmentInfo	TpCallAdditionalTreatmentInfo

TpCallTreatmentType

Defines the treatment for calls that will be handled only by the network.

Name	Value	Description
P_CALL_TREATMENT_DEFAULT	0	Default treatment
P_CALL_TREATMENT_RELEASE	1	Release the call
P_CALL_TREATMENT_SIAR	2	Send information to the user, and release the call (Send Info & Release)

TpCallAdditionalTreatmentInfo

Defines the Tagged Choice of Data Elements that specify the information to be sent to a call party.

	Tag Element Type	
	TpCallTreatmentType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_TREATMENT_DEFAULT	NULL	Undefined
P_CALL_TREATMENT_RELEASE	NULL	Undefined
P_CALL_TREATMENT_SIAR	TpUIInfo	InformationToSend

TpMediaType

Defines the media type of a media stream. The values may be combined by a logical 'OR' function.

Name	Value	Description
P_AUDIO	1	Audio stream
P_VIDEO	2	Video stream
P_DATA	4	Data stream (e.g., T120)

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010660**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **012** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to GCC STD
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

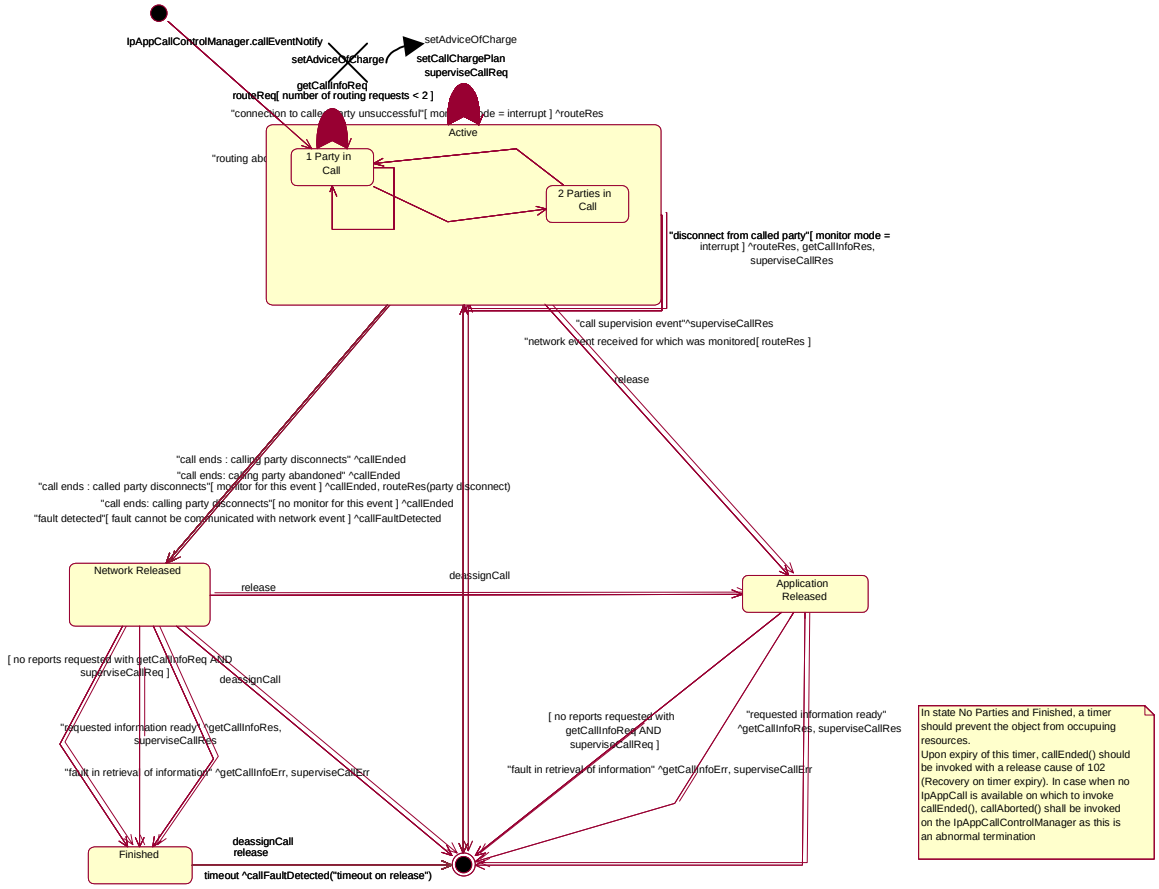
Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	setAdviceOfCharge was earlier erroneously specified as not possible in active state of call. This is unlike in R99. CAMEL supports advice of charge by SCI operation also during calls. See e.g. SendChargingInformation procedure description in TS 29.078 or gsmSSF description in TS 23.078.
Summary of change:	⌘	It is indicated in the STD that setAdviceOfCharge is possible in active state. This is reflected also in the text descriptions. Also a few errors in the text have been corrected and ETSI specification related extra text removed.
Consequences if not approved:	⌘	Erroneous specification.

Clauses affected:	⌘	4.1.1, 6.4.2
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

4.1.1 State Transition Diagrams for IpCall

The state transition diagram shows the application view on the Call object. This diagram shows only the part of the state transition diagram valid for 3GPP (UMTS) release 99.


IpAppCallControlManager.callEventNotify
setAdviceOfCharge
setAdviceOfCharge
setCallChargePlan
superviseCallReq
getCallInfoReq
routeReq[number of routing requests < 2]
"connection to called party unsuccessful"[monitor mode = interrupt] ^routeRes
Active
1 Party in Call
2 Parties in Call
"disconnect from called party"[monitor mode = interrupt] ^routeRes, getCallInfoRes, superviseCallRes
"call supervision event"^superviseCallRes
"network event received for which was monitored[routeRes]
release
"call ends : calling party disconnects" ^callEnded
"call ends: calling party abandoned" ^callEnded
"call ends : called party disconnects"[monitor for this event] ^callEnded, routeRes(party disconnect)
"call ends: calling party disconnects"[no monitor for this event] ^callEnded
"fault detected"[fault cannot be communicated with network event] ^callFaultDetected
Network Released
release
deassignCall
Application Released
[no reports requested with getCallInfoReq AND superviseCallReq]
deassignCall
"requested information ready" ^getCallInfoRes, superviseCallRes
"fault in retrieval of information" ^getCallInfoErr, superviseCallErr
[no reports requested with getCallInfoReq AND superviseCallReq]
"requested information ready" ^getCallInfoRes, superviseCallRes
"fault in retrieval of information" ^getCallInfoErr, superviseCallErr
Finished
deassignCall
release
timeout ^callFaultDetected("timeout on release")
In state No Parties and Finished, a timer should prevent the object from occupuing resources.
Upon expiry of this timer, callEnded() should be invoked with a release cause of 102 (Recovery on timer expiry). In case when no IpAppCall is available on which to invoke callEnded(), callAborted() shall be invoked on the IpAppCallControlManager as this is an abnormal termination

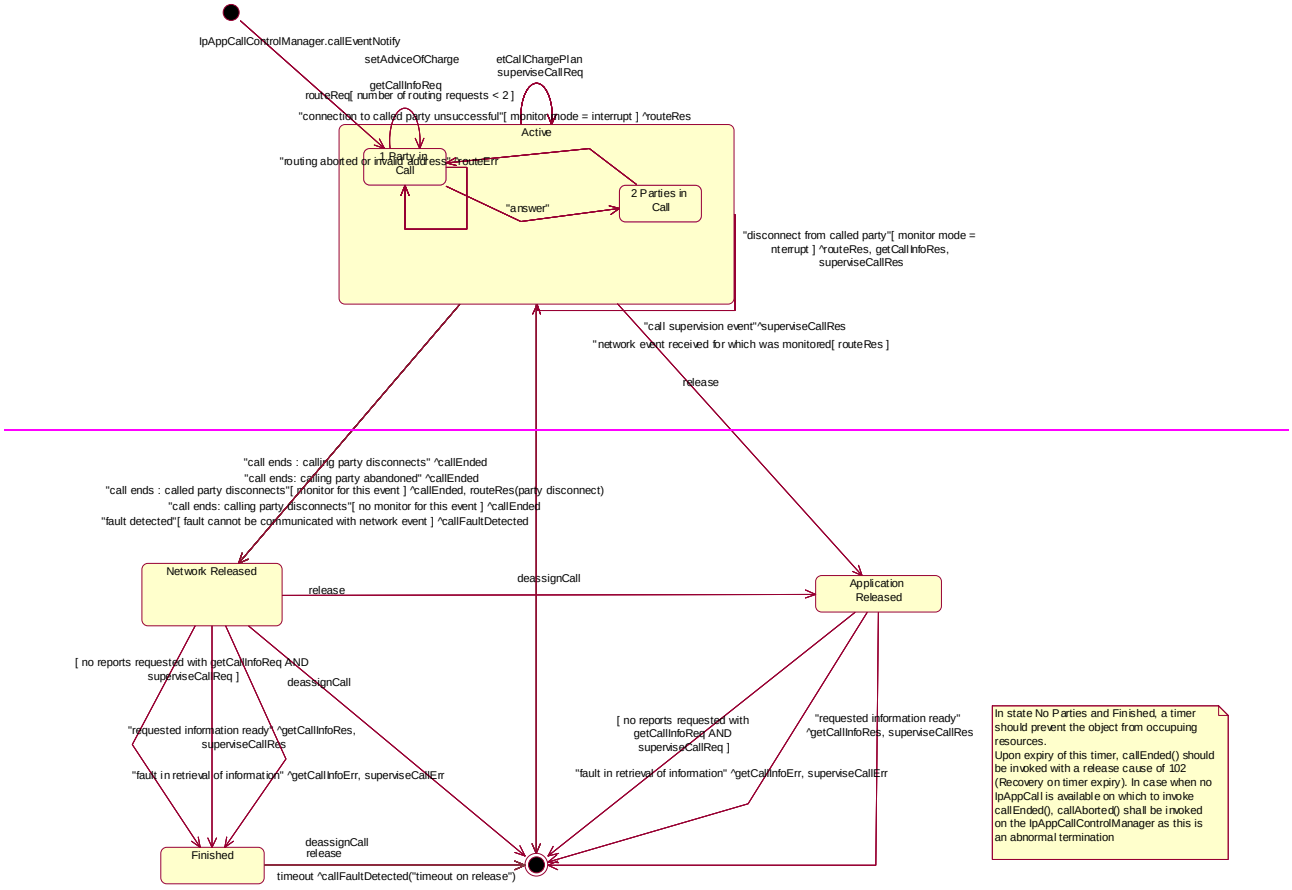


Figure : ~~3GPP~~Application view on the Call

4.1.1.1 Network Released State

In this state the call has ended and the Gateway collects the possible call information requested with getCallInfoReq() and / or superviseCallReq(). The information will be returned to the application by invoking the methods getCallInfoRes() and / or superviseCallRes() on the application. Also when a call was unsuccessful these methods are used.In case the application has not requested additional call related information immediately a transition is made to state Finished ~~No Parties~~.

4.1.1.2 Finished State

In this state the call has ended and no call related information is to be send to the application. The application can only release the call object. Calling the deassignCall() operation has the same effect. Note that the application has to release the object itself as good OO practice requires that when an object was created on behalf of a certain entity, this entity is also responsible for destroying it when the object is no longer needed.

4.1.1.3 Application Released State

In this state the application has requested to release the Call object and the Gateway collects the possilbe call information requested with getCallInfoReq() and / or superviseCallReq(). In case the application has not requested additional call related information the Call object is destroyed immediately.

~~6.4.2.4No Parties State~~

~~In this state the Call object has been created. The application can request the gateway for a certain type of charging of the call by calling setCallChargePlan(). The application can request for charging related information by calling~~

~~getCallInfoReq(). Furthermore the application can request supervision of the call by calling superviseCallReq(). It is also allowed to request Advice of Charge information to be sent by calling setAdviceOfCharge().~~

~~6.4.2.5~~4.1.1.4 Active State

In this state a call between two parties is being setup or present. Refer to the substates for more details. The application can request supervision of the call by calling superviseCallReq(). It is also allowed to send Advice of Charge information by calling setAdviceOfCharge() as well as to define the charging by invoking setCallChargePlan.

~~6.4.2.6~~4.1.1.5 1 Party in Call State

When the Call is in this state a calling party is present. The application can now request that a connection to a called party be established by calling the method routeReq().

In this state the application can also request the gateway for a certain type of charging of the call by calling setCallChargePlan(). The application can also request for charging related information by calling getCallInfoReq(). The setCallChargePlan() and getCallInfoReq() should be issued before requesting a connection to a called party by means of routeReq().

When the calling party abandons the call before the application has invoked the routeReq() operation, the gateway informs the application by invoking callFaultDetected() and also the operation callEnded() will be invoked. When the calling party abandons the call after the application has invoked routeReq() but before the call has actually been established, the gateway informs the application by invoking callEnded().

When the called~~ing~~ party answers the call, a transition will be made to the 2 Parties in Call state. In case the call can not be established because the application supplied an invalid address or the connection to the called party was unsuccessful while the application was monitoring for the latter in interrupt mode, the Call object will stay in this state

In this state user interaction is possible unless there is an outstanding routing request.

~~6.4.2.7~~4.1.1.6 2 Parties in Call State

A connection between two parties has been established.

In case the calling party disconnects, the gateway informs the application by invoking callEnded().

When the called party disconnects different situations apply:

1. the application is monitoring for this event in interrupt mode: a transition is made to the 1 Party in Call state, the application is informed with routeRes with indication that the called party has disconnected and all requested reports are sent to the application. The application now again has control of the call.

2. the application is monitoring for this event but not in interrupt mode. In this case a transition is made to the Network Released state and the gateway informs the application by invoking the operation routeRes() and callEnded().

3. the application is not monitoring for this event. In this case the application is informed by the gateway invoking the callEnded() operation and a transition is made to the Network Released state.

In this state user interaction is possible, depending on the underlying network.

~~6.4.2.8Routing to Destination(s) State~~

~~In this state there is at least one outstanding routeReq.~~

3GPP TSG_CN5 (Open Service Access – OSA) **Tdoc N5-010693**
Meeting #12

CR-Form-v3

CHANGE REQUEST

⌘ **29.198-04** **CR** **013** ⌘ rev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Introduction of sequence diagrams for MPCC services
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	In relation to the introduction of call leg STDs for MPCC new service example are demanded to clarify the usage of the MPCC API and its relationship to call leg STDs
Summary of change:	⌘	1) Modifications to the text describing the scope for MPCC Service to be more generic (text in chapter 7) 2) add text to the "application Initiated call setup" in chapter 7.1 to cater for a possible extension to a 3-party call service 3) Add new message sequence diagrams for 3 new MPCC service examples a) Call Information Collect service (new chapter 7.4.4) b) Hot-line service (new chapter 7.4.5) c) Call Forwarding on Busy service (new chapter 7.4.6)
Consequences if not approved:	⌘	Lacking clarification of the behaviour of MPCC API including the usage of MPCC Call Leg STDs

Clauses affected:	⌘	Section 7, 7.1 + new sections 7.4.4, 7.4.5 and 7.4.6.
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

Introduction

While introducing the MPCC call leg STDs for MPCC a need was identified to provide service examples to verify the behaviour if the call leg STDs and to show some use cases of the MPCC API.

Ericsson kindly requests the meeting to consider this change and approve it for inclusion into the MPCC API.

Proposed Changes

The proposed changes in this document are:

1) Modifications to the text handling the scope for MPCC Service text in chapter 7 and add text to the “application Initiated call setup” in chapter 7.1 to cater for a possible extension to a 3-party call service.

2) New message sequence diagrams for
   a) Call Information Collect service (new chapter 7.4.4)
   b) Hot-line service (new chapter 7.4.5)
   c) Call Forwarding on Busy service (new chapter 7.4.6)

See the proposed changes below (with revision marks for changed parts).

Proposed Modifications to chapter 7 and 7.1

7 MultiParty Call Control Service

The Multi-Party Call Control API ~~of 3GPP Rel4 relies on the CAMEL Service Environment (CSE).~~shields the complexity of the network, its protocols and specific implementation from the applications. This means that applications do not have to be aware of the network nodes a Service Capability Server (SCS) interacts with in order to provide the Multi Party Call Control service to the application. The specific underlying network and its protocols are transparent to the application.

NOTE: It should be noted that if the underlaying network is represented by CAMEL phase 3 a number of restrictions exist because CAMEL phase 3 supports only two-party calls and no leg based operations. Furthermore application initiated calls are not supported in CAMEL phase 3.

The detailed description of the supported methods is given in the chapter 7.5.

In some of the sequence diagrams the SCS is included to indicate the MPCC API relationship with the underlaying network by indicating the network events involved. The MPCC API relationship with SCS is implementation specific. The SCS is assumed to exist. Its representation in the sequence diagrams is not intended to imply any specific implementation.

7.1 Sequence Diagrams

7.1.1 Application initiated call setup

The following sequence diagram shows an application creating a call between party A and party B. Here, a call is created first. Then party A's call leg is created before events~~triggers~~ are requested ~~set~~ on it for answer and then routed to the call. On answer from party A, an announcement is played indicating that the call is being set up to party B. While the announcement is being played, party B's call leg is created and then events ~~triggers~~ are requested ~~set~~ on it for answer. On answer from party B the announcement is cancelled and party B is routed to the call.

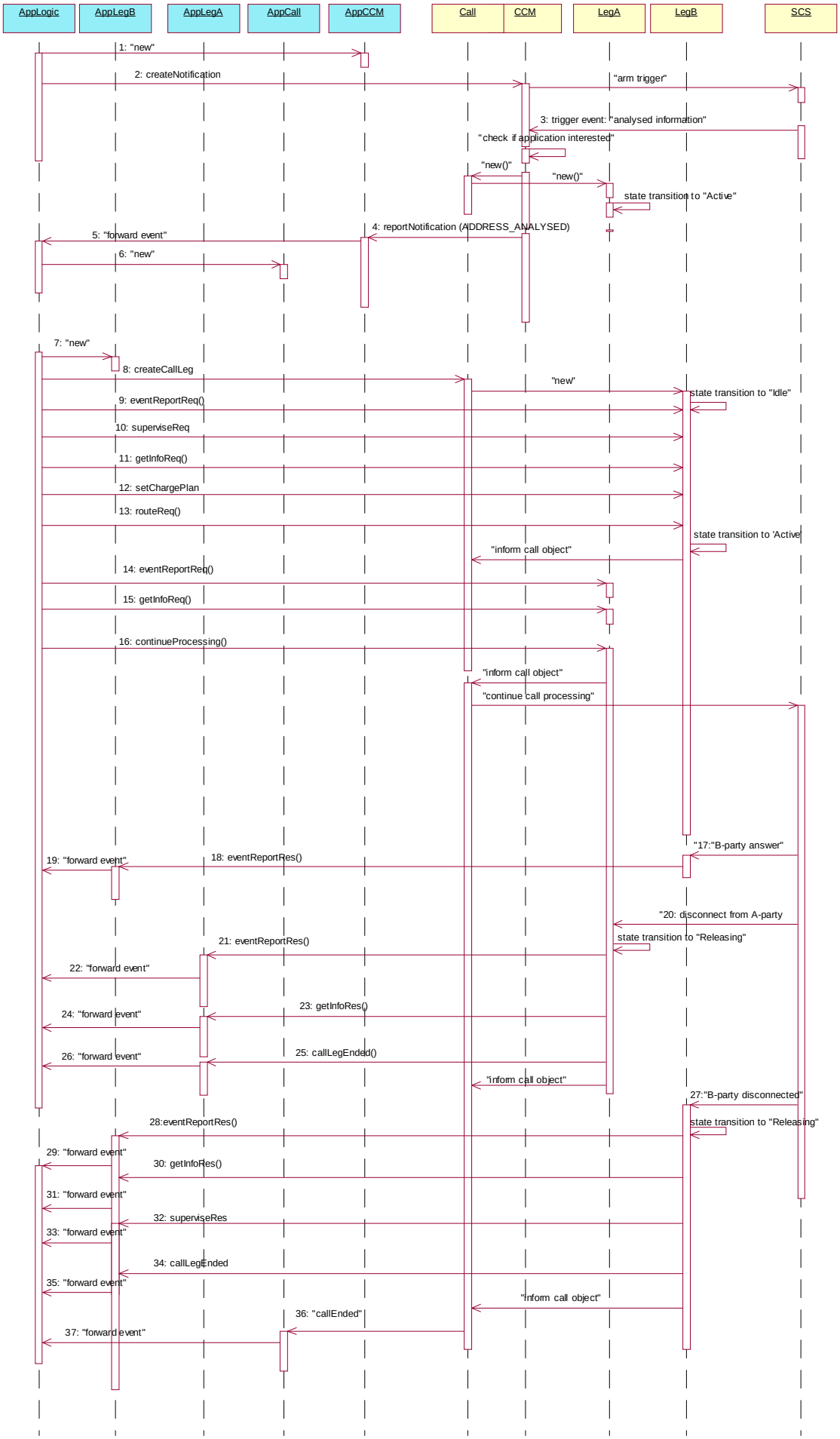
The service may as a variation be extended to include 3 parties (or more). After the two party call is established, the application can create a new leg and request to route it to a new destination address in order to establish a 3 party call. The event that causes this to happen could for example be the report of answer event from B-party or controlled by the A-party by entering a service code (mid-call event).
The procedure for call setup to party C is exactly the same as for the set up of the connection to party B (sequence 13 to 17 in the sequence diagram).

Proposed New Message Sequence Charts:

7.4.4 Call Information Collect service

The following sequence diagram shows an application monitoring a call between party A and a party B in order to collect call information at the end of the call for e.g. charging and/or statistic information collection purposes. The service may apply to ordinary two-party calls, but could also include a number translation of the dialed number and special charging (e.g. a premium rate service).

Additional call leg related information is requested with the getInfoReq and superviseReq methods.
The answer and call release events are in this service example requested to be reported in notify mode and additional call leg related information is requested with the getInfoReq and superviseReq methods in order to illustrate the information that can be collected and sent to the application at the end of the call.
Furthermore is shows the order in which information is sent to the application: network release event followed by possible requested call leg information, then thedestroy of the call leg object (callLegEnded method) and finally the destroy of the call object (callEnded).


AppLogic
AppLegB
AppLegA
AppCall
AppCCM
Call
CCM
LegA
LegB
SCS
1: "new"
2: createNotification
"arm trigger"
3: trigger event: "analysed information"
"check if application interested"
"new()"
"new()"
state transition to "Active"
4: reportNotification (ADDRESS_ANALYSED)
5: "forward event"
6: "new"
7: "new"
8: createCallLeg
"new"
state transition to "Idle"
9: eventReportReq()
10: superviseReq
11: getInfoReq()
12: setChargePlan
13: routeReq()
state transition to 'Active'
"inform call object"
14: eventReportReq()
15: getInfoReq()
16: continueProcessing()
"inform call object"
"continue call processing"
"17:"B-party answer"
18: eventReportRes()
19: "forward event"
"20: disconnect from A-party
state transition to "Releasing"
21: eventReportRes()
22: "forward event"
23: getInfoRes()
24: "forward event"
25: callLegEnded()
26: "forward event"
"inform call object"
27:"B-party disconnected"
state transition to "Releasing"
28:eventReportRes()
29: "forward event"
30: getInfoRes()
31: "forward event"
32: superviseRes
33: "forward event"
34: callLegEnded
35: "forward event"
"inform call object"
36: "callEnded"
37: "forward event"

1:This message is used by the application to create an object implementing the IpAppMultiPartyCallControlManager interface.

2:This message is sent by the application to enable notifications on new call events.

3: When a new call, that matches the event criteria, arrives a message (“analysed information”) is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface is met, other messages are used to create the call and associated call leg object.

4:This message is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface. Applied monitor mode is “interrupt”.

5: This message is used to forward message 4 to the IpAppLogic.

6: This message is used by the application to create an object implementing the IpAppMultiPartyCall interface. The reference to this object is passed back to the object implementing the IpMultiPartyCallControlManager using the return parameter of the callEventNotify.

7: A new AppCallLeg is created to receive callbacks for another leg..

8: This message is used to create a new call leg object. The object is created in the idle state and not yet routed in the network.

9: The application requests to be notified (monitor mode “NOTIFY”) when party B answers the call and when the leg to B-party is released.

10. The application requests to supervise the call leg to party B

11. The application requests information associated with the call leg to party b for example to calculate charging.

12. The application requests a specific charge plan to be set for the call leg to party B.

13: The application requests to route the terminating leg to reach the associated party B.

14: The application requests to be notified (monitor mode “NOTIFY”) when the leg to A-party is released.

15. The application requests to supervise the call leg to party A.

16: The application requests to resume call processing for the originating call leg.
As a result call processing is resumed in the network that will try to reach the associated party B.

17: When the B-party answers the call, the termination call leg is notified.

18:Assuming the call is answered, the object implementing party B's IpCallLeg interface passes the result of the call being answered back to its callback object (monitor mode “NOTIFY”).

19.This answer message is then forwarded to the object implementing the IpAppLogic interface.

20: When the A-party releases the call, the originating call leg is notified (monitor mode “NOTIFY”) and makes a transition to “releasing state”.

21: The application IpAppLegA is notified, as the release event has been requested to be reported in monitor mode “NOTIFY”.

22: The event is forwarded to the application logic.

23: The supervised call leg information is reported.

24: The event is forwarded to the application logic.

25: The origination call leg is destroyed, the AppLegA is notified

26: The event is forwarded to the application logic.

27: When the B-party releases the call or the call is released as a result of the release request from party A, i.e. a “originating release” indication, the terminating call leg is notified and makes a transition to “releasing state”.

28: If a network release event is received being a “terminating release” indication from called party B, the application IpAppLegB is notified, as the release event from party B has been requested to be reported in monitor mode “NOTIFY”.

Note: No report is sent if the release is caused by propagation of a network release event being an “originating release” indication coming from calling party A.

29: The event is forwarded to the application logic.

30: The supervised call leg information is reported.

31: The event is forwarded to the application logic.

32: The supervised call leg information is reported.

33: The event is forwarded to the application logic.

34: The terminating call leg is destroyed, the AppLegB is notified

35: The event is forwarded to the application logic.

36: When the originating call leg is destroyed, the AppLeg1 is notified

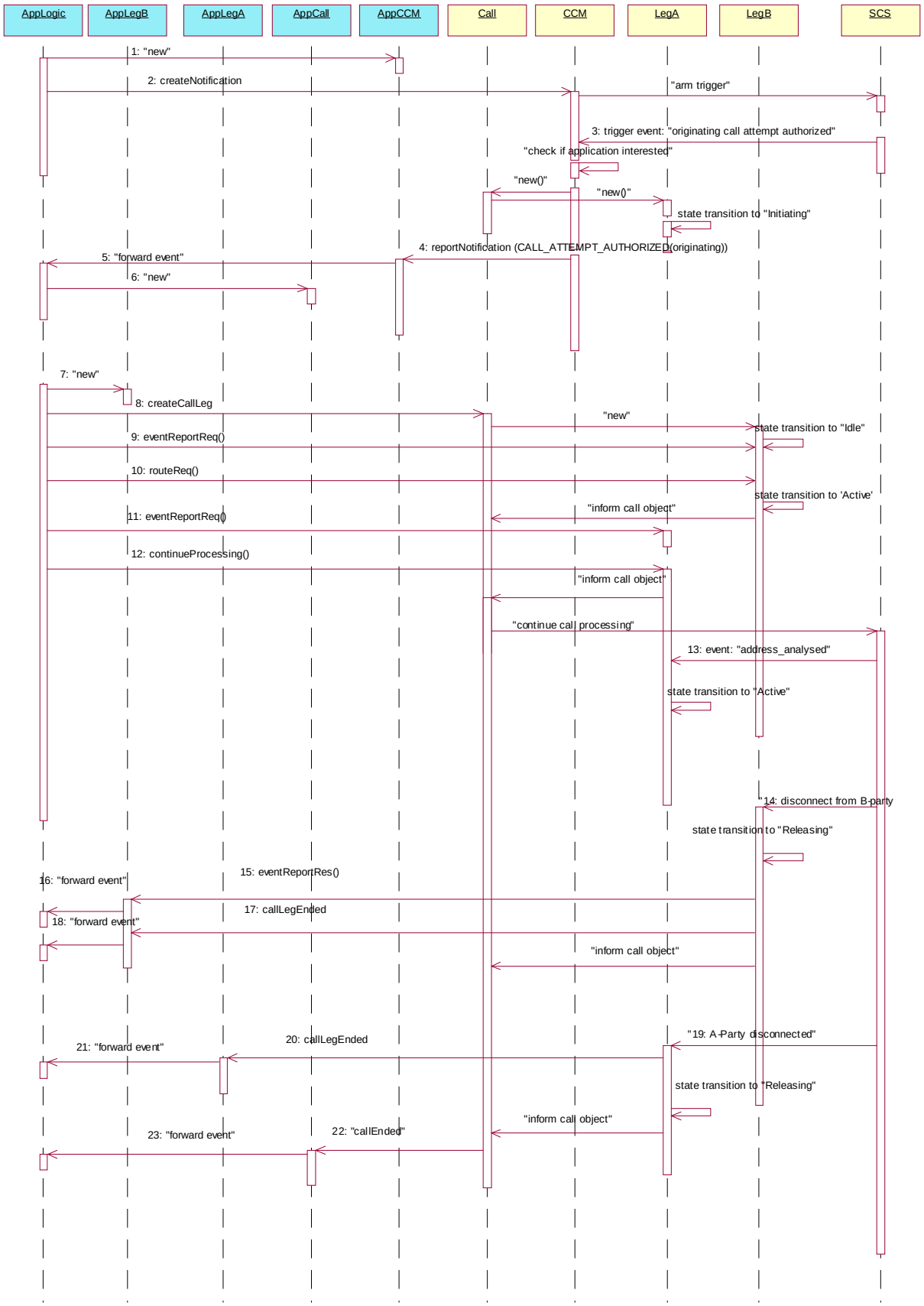
37: Assuming the IpCall object has been informed that the legs have been destroyed, the the IpAppMultiPartyCall is notified that the call is ended.

38: The event is forwarded to the application logic.

7.4.5 Hot-Line service

The following sequence diagram shows an application establishing a call between party A and pre-arranged party B defined to constitute a hot-line address. The address of the destination party is provided by the application as the calling party makes a call attempt (goes off-hook) and do not dial any number within a predefined time. In this case a pre-defined number (hot-line number) is provided by the application. The call is then routed to the pre-defined destination party.
The call release is monitored to enable the sending of information to the application at call release, e.g. for charging purposes.

Note: This service could be extended as follows:
Sometime during the call the calling party enters '#5' which causes the called leg to be released. The calling party is now prompted to enter the address of a new destination party, to which it is then routed.
This scenario is handled in 7.4.3.


AppLogic
AppLegB
AppLegA
AppCall
AppCCM
Call
CCM
LegA
LegB
SCS
1: "new"
2: createNotification
"arm trigger"
3: trigger event: "originating call attempt authorized"
"check if application interested"
"new()"
"new()"
state transition to "Initiating"
4: reportNotification (CALL_ATTEMPT_AUTHORIZED(originating))
5: "forward event"
6: "new"
7: "new"
8: createCallLeg
"new"
state transition to "Idle"
9: eventReportReq()
10: routeReq()
state transition to 'Active'
"inform call object"
11: eventReportReq()
12: continueProcessing()
"inform call object"
"continue call processing"
13: event: "address_analysed"
state transition to "Active"
"14: disconnect from B-party
state transition to "Releasing"
15: eventReportRes()
16: "forward event"
17: callLegEnded
18: "forward event"
"inform call object"
"19: A-Party disconnected"
20: callLegEnded
21: "forward event"
state transition to "Releasing"
"inform call object"
22: "callEnded"
23: "forward event"

1:This message is used by the application to create an object implementing the IpAppMultiPartyCallControlManager interface.

2:This message is sent by the application to enable notifications on new call events.

3: When a new call, that matches the event criteria, arrives a message (originating call attempt" is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface is met, other messages are used to create the call and associated call leg object.

4:This message (monitor mode "INTERRUPT") is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface.

5: This message is used to forward message 4 to the IpAppLogic.

6: This message is used by the application to create an object implementing the IpAppMultiPartyCall interface. The reference to this object is passed back to the object implementing the IpMultiPartyCallControlManager using the return parameter of the callEventNotify.

7: A new AppCallLegB is created to receive callbacks for another leg..

8: This message is used to create a new call leg object. The object is created in the idle state and not yet routed in the network.

9: The application requests to be notified (monitor mode "NOTIFY") when the leg to party B is released.

10: The application requests to route the terminating leg to reach the associated party as specified by the application ("hot-line number").

11: The application requests to be notified (monitor mode "NOTIFY") when the leg to party A is released.

12: The application requests to resume call processing for the originating call leg.
As a result call processing is resumed in the network that will try to reach the associated party as specified by the application (E.164 number provided by application)

13: The originating call leg is notified that the number (provided by application) has been analysed by the network and the originating call leg STD makes a transition to "active" state. The application is not notified as it has not requested this event to be reported.

14: When the B-party releases the call, the terminating call leg is notified (monitor mode "NOTIFY") and makes a transition to "Releasing state".

15: The application is notified, as the release event has been requested to be reported in Notify mode..

16: The event is forwarded to the application logic.

17: When the terminating call leg is destroyed, the AppLegB is notified

18: The event is forwarded to the application logic.

19: When the call release ("terminating release" indication) is propagated in the network toward the party A, the originating call leg is notified and makes a transition to "releasing state". This release event (being propagated from party B) is not reported to the application.

20: When the originating call leg is destroyed, the AppLegA is notified

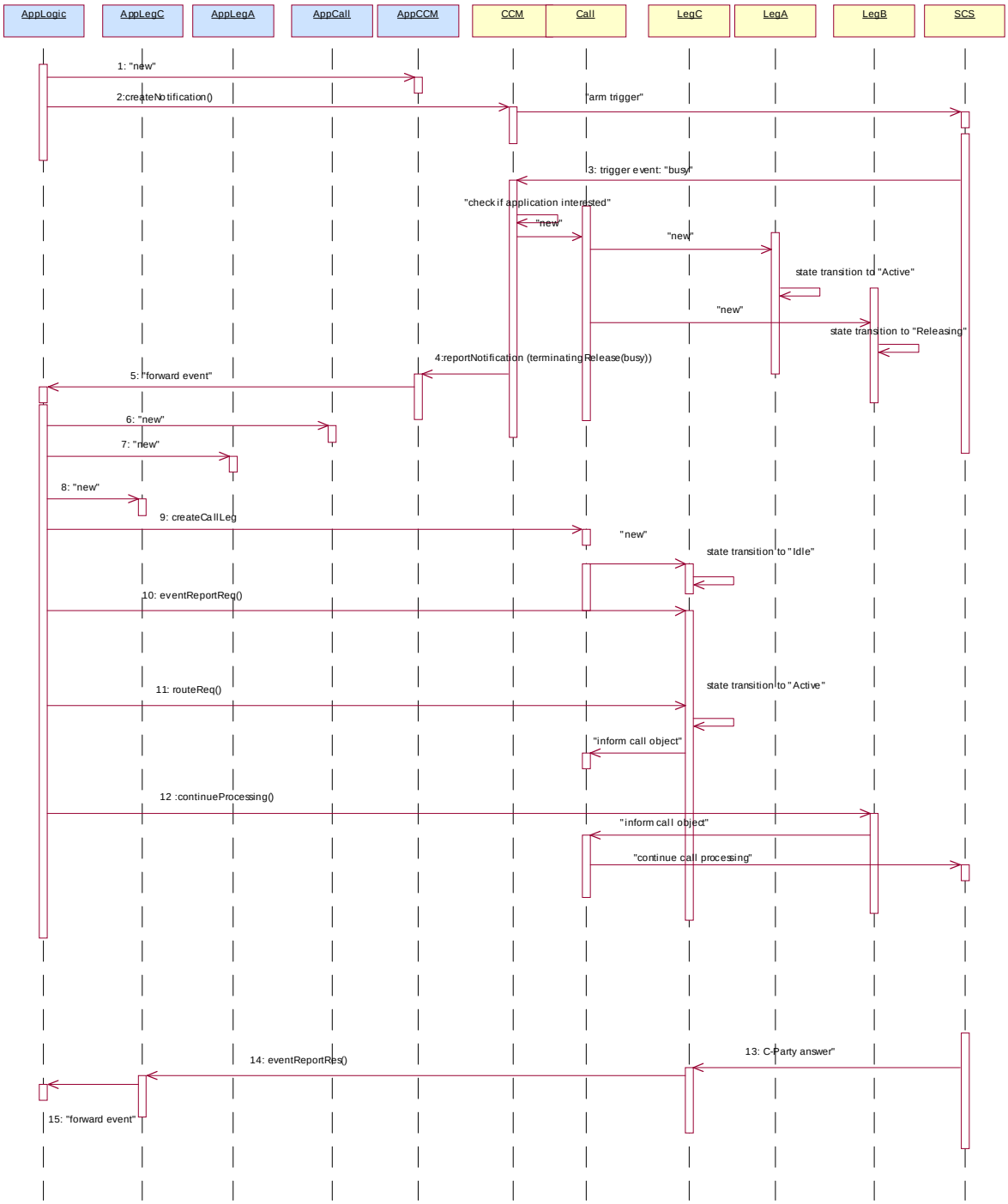
21: The event is forwarded to the application logic.

22: When all legs have been destroyed, the IpAppMultiPartyCall is notified that the call is ended.

23: The event is forwarded to the application logic.

7.4.6 Call Forwarding on Busy service

The following sequence diagram shows an application establishing a call forwarding on busy.
When a call is made from A to B but the B-party is detected to be busy, then the application is informed of this and sets up a connection towards a C party. The C party can for instance be a voicemail system.



1:This message is used by the application to create an object implementing the IpAppMultiPartyCallControlManager interface.

2:This message is sent by the application to enable notifications on new call events.

3: When a new call, that matches the event criteria, arrives a message (“busy”) is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface is met, other messages are used to create the call and associated call leg objects.

4:This message is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface. (monitor mode “INTERRUPT”).

5: This message is used to forward message 4 to the IpAppLogic.

6: This message is used by the application to create an object implementing the IpAppMultiPartyCall interface. The reference to this object is passed back to the object implementing the IpMultiPartyCallControlManager using the return parameter of the callEventNotify.

7: A new AppCallLegA is created to receive callbacks for another leg..

8: A new AppCallLegC is created to receive callbacks for another leg..

9: This message is used to create a new call leg object. The object is created in the idle state and not yet routed in the network.

10: The application requests to be notified (monitor mode “INTERRUPT”) when party C answers the call..

11: The application requests to route the terminating leg to reach the associated party C.
The application may request if so desired a call redirection by including the original destination address (field P_CALL_APP_ORIGINAL_DESTINATION_ADDRESS of TpCallAppInfo) in the request to route the call leg to the remote party C.

12: The application requests to resume call processing for the terminating call leg to party B to terminate the leg. Alternative the application could request to deassign the leg to party B for example if it is not interested in possible requested call leg information (getInfoRes, superviseRes).
When the terminating call leg is destroyed, the AppLegB is notified and the event is forwarded to the application logic (not shown). As a result call processing is resumed in the network that will try to reach the associated party C.

13: When the party C answers the call, the termination call leg is notified (monitor mode “NOTIFY”).

14:Assuming the call is answered, the object implementing party C's IpCallLeg interface passes the result of the call being answered back to its callback object.

15.This answer message is then forwarded to the object implementing the IpAppLogic interface.

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010707**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **014** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	The use of the REDIRECT event needs to be illustrated
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	CN5 believes that the REDIRECTED event is vital for detecting network-driven call forwarding. However, its use is not particularly intuitive and it is unclear what this event means and why it would be used.
Summary of change:	⌘	It is proposed to add a sequence diagram for MPCC, illustrating the use of the REDIRECTED event. It is also proposed that the REDIRECTED event should not be disarmed once it has fired. This is because the call could be redirected more than once, and it does not seem appropriate that the application should have to register for the REDIRECTED event in INTERRUPT mode, simply so that it can re-arm that event once it has fired (it is similar to the QUEUED event in this respect).
Consequences if not approved:	⌘	29.198-4 will be ambiguous and difficult to implement correctly – interworking will be jeopardised. Failure to adopt this CR would result in divergence between the 3GPP R4 specification and the ETSI/Parlay specifications.

Clauses affected:	⌘	7.1.4, 7.6.2
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at:
http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

Problem

Lucent believes that the REDIRECTED event is vital for detecting network-driven call forwarding. However, its use is not particularly intuitive and it is unclear what this event means.

When an application routes a call leg, the events it receives are for the leg, not for the destination address of that leg. We believe that the REDIRECTED event will be sent to the application if network call forwarding has been detected. The new destination address is sent along with the event. Subsequent events received for that leg are events relating to the new destination address.

Proposal

Lucent proposes to add a sequence diagram for MPCC, illustrating the use of the REDIRECTED event.

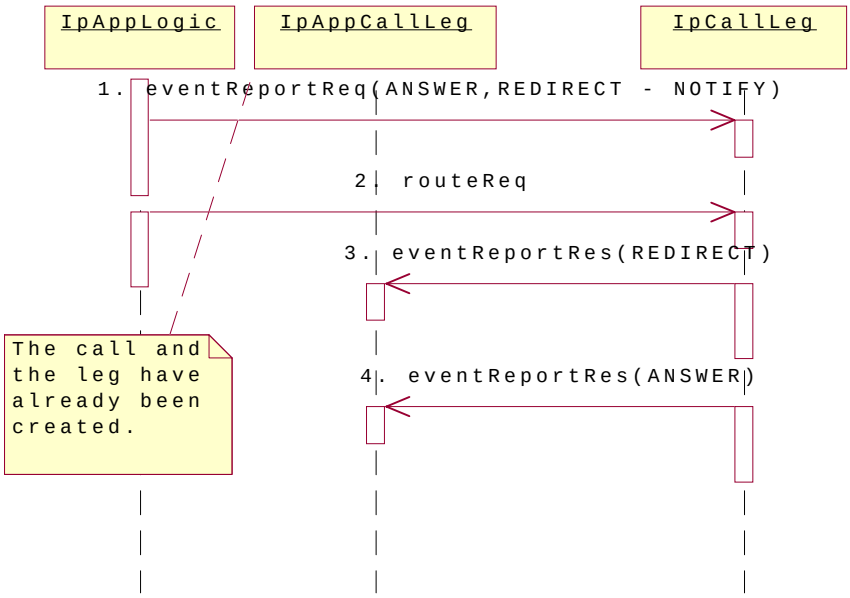
Lucent also proposes that the REDIRECTED event should not be disarmed once it has fired. This is because the call could be redirected more than once, and it does not seem appropriate that the application should have to register for the REDIRECTED event in INTERRUPT mode, simply so that it can re-arm that event once it has fired (it is similar to the QUEUED event in this respect).

Lucent has noticed that the PROGRESS event has been deleted from the event table but is still present in the table that specifies the disarming rules. We have proposed deleting this.

Resulting changes

Lucent would like to add the following sequence diagram to section 7.1

7.1.4 Network-initiated Call Redirection



1. The application has already created the call and a call leg. It places an event report request for the ANSWER and REDIRECTED events in NOTIFY mode.

2. The application routes the call leg.

3. The call is redirected within the network and the application is informed. The new destination address is passed within the event. The event is not disarmed, so subsequent redirections will also be reported. Also, the same call leg is used so the application does not have to create a new one.

4. The call is answered at its new destination.

7.6.2 Multi-Party Call Control Data Definitions

TpCallEventType

Defines a specific call event report type.

Name	Value	Description
P_CALL_EVENT_UNDEFINED	0	Undefined
P_CALL_EVENT_CALL_ATTEMPT	1	A Call attempt takes place (e.g. Offhook event)
P_CALL_EVENT_ADDRESS_COLLECTED	2	The destination address has been collected
P_CALL_EVENT_ADDRESS_ANALYSED	3	The destination address has been analysed
P_CALL_EVENT_ALERTING	5	Call is alerting at the call party
P_CALL_EVENT_ANSWER	6	Call answered at address
P_CALL_EVENT_RELEASE	7	A Call has been released or the call could not be routed
P_CALL_EVENT_REDIRECTED	8	Call redirected to new address: an indication from the network that the call has been redirected to a new address. (No events are disarmed as a result of this)
P_CALL_EVENT_SERVICE_CODE	9	Mid-call service code received
P_CALL_EVENT_QUEUED	10	The Call Event has been queued. (no events are disarmed as a result of this)

The table below defines the disarming rules for dynamic events. In case such an event occurs the table shows which events are disarmed (are not monitored anymore) and should be re-armed by eventReportReq() in case the application is still interested in these events.

Event Occured	Events Disarmed
P_CALL_EVENT_UNDEFINED	Not Applicable
P_CALL_EVENT_CALL_ATTEMPT	Not applicable, can only be armed as trigger
P_CALL_EVENT_ADDRESS_COLLECTED	P_CALL_EVENT_ADDRESS_COLLECTED
P_CALL_EVENT_ADDRESS_ANALYSED	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED
~~P_CALL_EVENT_PROGRESS~~	~~P_CALL_EVENT_ADDRESS_COLLECTED~~ ~~P_CALL_EVENT_ADDRESS_ANALYSED~~ ~~P_CALL_EVENT_PROGRESS~~
P_CALL_EVENT_ALERTING	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED P_CALL_EVENT_PROGRESS P_CALL_EVENT_ALERTING P_CALL_EVENT_RELEASE with criteria: • P_USER_NOT_AVAILABLE • P_BUSY • P_NOT_REACHABLE • P_ROUTING_FAILURE • P_CALL_RESTRICTED • P_UNAVAILABLE_RESOURCES
P_CALL_EVENT_ANSWER	P_CALL_EVENT_ADDRESS_COLLECTED P_CALL_EVENT_ADDRESS_ANALYSED P_CALL_EVENT_PROGRESS P_CALL_EVENT_ALERTING P_CALL_EVENT_RELEASE with criteria:

	• P_USER_NOT_AVAILABLE • P_BUSY • P_NOT_REACHABLE • P_ROUTING_FAILURE • P_CALL_RESTRICTED • P_UNAVAILABLE_RESOURCES • P_NO_ANSWER • P_PREMATURE_DISCONNECT P_CALL_EVENT_ANSWER
P_CALL_EVENT_RELEASE	All pending events are disarmed
~~P_CALL_EVENT_REDIRECTED~~	~~P_CALL_EVENT_REDIRECTED~~
P_CALL_EVENT_SERVICE_CODE	P_CALL_EVENT_SERVICE_CODE

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010709**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **015** ⌘ ev **-** ⌘ Current version: **4.0.0.** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to SetCallChargePlan()
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	We noted some technical flaws and corrections on the SetCallChargePlan() methods for IpCall
Summary of change:	⌘	Parameters TpCallChargePlan and TpCallChargeOrderCategory needs corrections.
Consequences if not approved:	⌘	Correction of technical flaws needs resolution.

Clauses affected:	⌘	6.6.1 for SetCallChargePlan() methods in Section 6.3.3.
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1 Introduction

It is proposed that the following technical flaws and corrections on the SetCallChargePlan() method parameters are introduced in the Draft ETSI 201 915-4.

2 Proposed corrections

The following parameters in the setCallChargePlan() methods of the Interface Class IpCall (Section 6.3.3) need to be corrected into section 6.6.1.

TpCallChargePlan

Defines the `Sequence of Data Elements` that specify the charge plan for the call.

Sequence Element Name	Sequence Element Type	Description
`ChargeOrderType`	`TpCallChargeOrderCategory`	Type of charging to be performed: time based charging or transparent charging or pre-defined charge plan.
~~`ChargePerTime`~~	~~`TpChargePerTime`~~	~~Charge per time.~~ ~~Only applicable when time based charging is selected.~~
`TransparentCharge`	`TpOctetSet`	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records. Only applicable when transparent charging is selected.
`ChargePlan`	`TpInt32`	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user). Only applicable when predefined~~transparent~~ charging is selected.
~~`Currency`~~	~~`TpString`~~	~~Currency unit according to ISO-4217:1995~~
`AdditionalInfo`	`TpOctetSet`	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
`PartyToCharge`	`TpCallPartyToCharge`	Party to be charged.

TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
~~`P_CALL_CHARGE_PER_TIME`~~	~~`0`~~	~~Charge per time~~
`P_CALL_CHARGE_TRANSPARENT`	0~~1~~	Operator specific charge plan specification, e.g. charging table name / charging table entry. The associated charge plan data will be send transparently to the charging records
`P_CALL_CHARGE_PREDEFINED_SET`	1~~2~~	Pre-defined charge plan. Example of the charge plan set from which the application can choose could be : (0 = normal user, 1 = silver card user, 2 = gold card user).

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010710**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **016** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Add one additional error indication
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	At the moment there are few error indications that can be returned for asynchronous methods using parameter of type TpCallError.
Summary of change:	⌘	Introduction of one additional error indication: unavailable resources in the SCS or the core network.
Consequences if not approved:	⌘	Applications cannot be informed of the fact that no resources are available to handle the request. Furthermore, it will lead to misalignment with ETSI ES 201 915 where the change was already accepted.

Clauses affected:	⌘	8
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

TpCallError

Defines the Sequence of Data Elements that specify the additional information relating to acall error.

Sequence Element Name	Sequence Element Type
ErrorTime	TpDateAndTime
ErrorType	TpCallErrorType
AdditionalErrorInfo	TpCallAdditionalErrorInfo

TpCallAdditionalErrorInfo

Defines the Tagged Choice of Data Elements that specify additional call error and call error specific information. This is also used to specify call leg errors and information errors.

	Tag Element Type	
	TpCallErrorType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CALL_ERROR_UNDEFINED	NULL	Undefined
P_CALL_ERROR_INVALID_ADDRESS	TpAddressError	CallErrorInvalidAddress
P_CALL_ERROR_INVALID_STATE	NULL	Undefined
P_CALL_ERROR_RESOURCE_UNAVAILABLE	NULL	Undefined

TpCallErrorType

Defines a specific call error.

Name	Value	Description
P_CALL_ERROR_UNDEFINED	0	Undefined; the method failed or was refused, but no specific reason can be given.
P_CALL_ERROR_INVALID_ADDRESS	1	The operation failed because an invalid address was given
P_CALL_ERROR_INVALID_STATE	2	The call was not in a valid state for the requested operation
P_CALL_ERROR_RESOURCE_UNAVAILABLE	3	There are not enough resources to complete the request successfully

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010663**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **017** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to Call Control – GCCS Exception handling
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	Exception handling mechanism for Generic Call Control Service requires correction to enable it to be correctly used, and to be consistent with exception handling for all other services and framework in Release 4.
Summary of change:	⌘	Replace TpGCCSExeption, TpGeneralException, with TpCommonExceptions and detailed exception classes which can be thrown for each method;
Consequences if not approved:	⌘	Release 4 GCCS is no longer in step with remainder of the Release 4 APIs and as such the application programmer is presented with a mixture of APIs as part of the single release. No new or modified GCCS functionality is required rather a consistent maintenance of GCCS as part of the overall API set is necessary. Failure to implement the agreed change shall result in divergence between 3GPP Release 4 GCCS and the service in ETSI/Parlay specifications.

Clauses affected:	⌘	6.3		
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	29.198-2 Common Data (CR29.198-02-004 N5-010658) presented also to CN#13 meeting
Other comments:	⌘	29.198-2 Common Data updated to remove TpResult and prior GCCS exception types (CR29.198-02-004_ N5-010658).		

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

6.3 Generic Call Control Service Interface Classes

The Generic Call Control Service (GCCS) provides the basic call control service for the API. It is based around a third party model, which allows calls to be instantiated from the network and routed through the network.

The GCCS supports enough functionality to allow call routing and call management for today's Intelligent Network (IN) services in the case of a switched telephony network, or equivalent for packet based networks.

It is the intention of the GCCS that it could be readily specialised into call control specifications, for example, ITU-T recommendations H.323, ISUP, Q.931 and Q.2931, ATM Forum specification UNI3.1 and the IETF Session Initiation Protocol, or any other call control technology.

The adopted call model has the following objects. Note that not all of these concepts are used in the generic call.

* a call object. A call is a relation between a number of parties. The call object relates to the entire call view from the application. E.g., the entire call will be released when a release is called on the call. Note that different applications can have different views on the same physical call, e.g., one application for the originating side and another application for the terminating side. The applications will not be aware of each other, all 'communication' between the applications will be by means of network signalling. The API currently does not specify any feature interaction mechanisms.

* a call leg object. The leg object represents a logical association between a call and an address. The relationship includes at least the signalling relation with the party. The relation with the address is only made when the leg is routed. Before that the leg object is IDLE and not yet associated with the address.

* an address. The address logically represents a party in the call.

* a terminal. A terminal is the end-point of the signalling and/or media for a party. This object type is currently not addressed.

The call object is used to establish a relation between a number of parties by creating a leg for each party within the call.

Associated with the signalling relationship represented by the call leg, there may also be a bearer connection (e.g., in the traditional voice only networks) or a number (zero or more) of media channels (in multi-media networks).

A leg can be attached to the call or detached from the call. When the leg is attached, this means that media or bearer channels related to the legs are connected to the media or bearer channels of the other legs that are attached to the same call. I.e., only legs that are attached can 'speak' to each other. A leg can have a number of states, depending on the signalling received from or sent to the party associated with the leg. Usually there is a limit to the number of legs that are in being routed (i.e., the connection is being established) or connected to the call (i.e., the connection is established). Also, there usually is a limit to the number of legs that can be simultaneously attached to the same call.

Some networks distinguish between controlling and passive legs. By definition the call will be released when the controlling leg is released. All other legs are called passive legs. There can be at most one controlling leg per call. However, there is currently no way the application can influence whether a Leg is controlling or not.

There are two ways for an application to get the control of a call. The application can request to be notified of calls that meet certain criteria. When a call occurs in the network that meets these criteria, the application is notified and can control the call. Some legs will already be associated with the call in this case. Another way is to create a new call from the application.

For the generic call control service, only a subset of the model is used; the API for generic call control does not give explicit access to the legs and the media channels. This is provided by the Multi-Party Call Control Service. Furthermore, the generic call is restricted to two party calls, i.e., only two legs are active at any given time. Active is defined here as 'being routed' or connected.

The GCCS is represented by the IpCallManager and IpCall interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppCallManager and IpAppCall to provide the callback mechanism.

6.3.1 Interface Class IpCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Generic Call Control Service. The generic call control manager interface provides the management functions to the generic call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications.

<<Interface>> IpCallControlManager
createCall (appCall : in IpAppCallRef, callReference : out TpCallIdentifierRef) : TpResult enableCallNotification (appCallControlManager : in IpAppCallControlManagerRef, eventCriteria : in TpCallEventCriteria, assignmentID : out TpAssignmentIDRef) : TpResult disableCallNotification (assignmentID : in TpAssignmentID) : TpResult setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange, assignmentID : out TpAssignmentIDRef) : TpResult changeCallNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpCallEventCriteria) : TpResult getCriteria (eventCriteria : out TpCallEventCriteriaResultSetRef) : TpResult

Method

createCall()

This method is used to create a new call object. An IpAppCallControlManager should already have been passed to the IpCallControlManager, otherwise the call control will not be able to report a callAborted()

to the application (the application should invoke setCallback() if it wishes to ensure this).

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

callReference : out TpCallIdentifierRef

Specifies the interface reference and sessionID of the call created.

Raises

~~**TpGCCSException, TpGeneralException**~~**TpCommonExceptions**

Method
enableCallNotification()

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notification of calls happening in the network. When such an event happens, the application will be informed by callEventNotify(). In case the application is interested in other events during the context of a particular call session it has to use the routeReq() method on the call object. The application will get access to the call object when it receives the callEventNotify(). (Note that the enableCallNotification() is not applicable if the call is setup by the application).

The enableCallNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P~~_GCCS~~_INVALID_CRITERIA.The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same CallNotificationType is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallBack().

Parameters

appCallControlManager : in IpAppCallControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

eventCriteria : in TpCallEventCriteria

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

assignmentID : out TpAssignmentIDRef

Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Raises

~~TpGCCSException,TpGeneralException~~TpCommonExceptions,P_INVALID_CRITERIA,P_INVALID_INTERFACE_TYPE,P_INVALID_EVENT_TYPE

Method
disableCallNotification()

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

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the generic call control manager interface when the previous enableNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENTID. If two callbacks have been registered under this assignment ID both of them will be disabled.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_ASSIGNMENT_ID**

Method

setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the generic call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

Parameters

duration : in TpDuration

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

mechanism : in TpCallLoadControlMechanism

Specifies the load control mechanism to use (for example, admit one call per interval), and any necessary parameters, such as the call admission rate. The contents of this parameter are ignored if the load control duration is set to zero.

treatment : in TpCallTreatment

Specifies the treatment of calls that are not admitted. The contents of this parameter are ignored if the load control duration is set to zero.

addressRange : in TpAddressRange

Specifies the address or address range to which the overload control should be applied or removed.

assignmentID : out TpAssignmentIDRef

Specifies the assignmentID assigned by the gateway to this request. This assignementID can be used to correlate the callOverlloadEncountered and callOverloadCeased methods with the request.

Raises

~~**TpGeneralException,TpGCCSException**~~**TpCommonExceptions,P_INVALID_ADDRESS,P_UNSUPPORTED_ADDRESS_PLAN**

Method

changeCallNotification()

This method is used by the application to change the event criteria introduced with enableCallNotification. Any stored criteria associated with the specified assignementID will be replaced with the specified criteria.

Parameters

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic call control manager interface for the event notification. If two call backs have been registered under this assignment ID both of them will be changed.

eventCriteria : in TpCallEventCriteria

Specifies the new set of event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

~~**TpGeneralException,TpGCCSException**~~**TpCommonExceptions,P_INVALID_ASSIGNMENT_ID,P_INVALID_CRITERIA,P_INVALID_EVENT_TYPE**

Method

getCriteria()

This method is used by the application to query the event criteria set with enableCallNotification or changeCallNotification.

Parameters

eventCriteria : out TpCallEventCriteriaResultSetRef

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

~~**TpGeneralException,TpGCCSException**~~**TpCommonExceptions**

6.3.2 Interface Class IpAppCallControlManager

Inherits from: IpInterface

The generic call control manager application interface provides the application call control management functions to the generic call control service.

<<Interface>> IpAppCallControlManager
callAborted (callReference : in TpSessionID) : TpResult callEventNotify (callReference : in TpCallIdentifier, eventInfo : in TpCallEventInfo, assignmentID : in TpAssignmentID, appCall : out IpAppCallRefRef) : TpResult callNotificationInterrupted () : TpResult

callNotificationContinued () : TpResult

callOverloadEncountered (assignmentID : in TpAssignmentID) : TpResult

callOverloadCeased (assignmentID : in TpAssignmentID) : TpResult

Method

callAborted()

This method indicates to the application that the call object (at the gateway) has aborted or terminated abnormally. No further communication will be possible between the call and application.

Parameters

callReference : in TpSessionID

Specifies the sessionID of call that has aborted or terminated abnormally.

~~*Raises*~~

~~**TpGCCSException,TpGeneralException**~~

Method

callEventNotify()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

When this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPT, the application writer should ensure that no routeReq() is performed until an IpAppCall has been passed to the gateway, either through an explicit setCallback() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

Parameters

callReference : in TpCallIdentifier

Specifies the reference to the call interface to which the notification relates. This parameter will be null if the notification is in NOTIFY mode.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

Specifies the assignment id which was returned by the enableNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

appCall : out IpAppCallRefRef

Specifies a reference to the application interface which implements the callback interface for the new call. This parameter will be null if the notification is in NOTIFY mode.

~~Raises~~

~~TpGCCSException,TpGeneralException~~

Method
callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporary interrupted (for example, due to faults detected).

Note that more permanent failures are reported via the Framework (integrity management).

Parameters
No Parameters were identified for this method

~~Raises~~

~~TpGCCSException,TpGeneralException~~

Method
callNotificationContinued()

This method indicates to the application that event notifications will again be possible.

Parameters
No Parameters were identified for this method

Method
callOverloadEncountered()

This method indicates 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.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been encountered.

~~Raises~~

~~TpGeneralException,TpGCCSException~~

Method
callOverloadCeased()

This method indicates 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.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been ceased

~~*Raises*~~

~~**TpGeneralException, TpGCCSException**~~

6.3.3 Interface Class IpCall

Inherits from: IpService

The generic Call provides the possibility to control the call routing, to request information from the call, control the charging of the call, to release the call and to supervise the call. It does not give the possibility to control the legs directly and it does not allow control over the media. The first capability is provided by the multi-party call and the latter as well by the multi-media call. The call is limited to two party calls, although it is possible to provide 'follow-on' calls, meaning that the call can be rerouted after the terminating party has disconnected or routing to the terminating party has failed. Basically, this means that at most two legs can be in connected or routing state at any time.

<<Interface>> IpCall
routeReq (callSessionID : in TpSessionID, responseRequested : in TpCallReportRequestSet, targetAddress : in TpAddress, originatingAddress : in TpAddress, originalDestinationAddress : in TpAddress, redirectingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, callLegSessionID : out TpSessionIDRef) : TpResult release (callSessionID : in TpSessionID, cause : in TpCallReleaseCause) : TpResult deassignCall (callSessionID : in TpSessionID) : TpResult getCallInfoReq (callSessionID : in TpSessionID, callInfoRequested : in TpCallInfoType) : TpResult setCallChargePlan (callSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : TpResult setAdviceOfCharge (callSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : TpResult getMoreDialledDigitsReq (callSessionID : in TpSessionID, length : in TpInt32) : TpResult superviseCallReq (callSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : TpResult

Method

routeReq()

This asynchronous method requests routing of the call to the remote party indicated by the targetAddress.

The extra address information such as originatingAddress is optional. If not present (i.e., the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in corresponding addresses from the route is used, otherwise the network or gateway provided numbers will be used.

If this method in invoked, and call reports have been requested, yet no IpAppCall interface has been provided, this method shall throw the P_NO_CALLBACK_ADDRESS_SET exception.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

responseRequested : in TpCallReportRequestSet

Specifies the set of observed events that will result in zero or more routeRes() being generated.

E.g., when both answer and disconnect is monitored the result can be received two times.

If the application wants to control the call (in whatever sense) it shall enable event reports

targetAddress : in TpAddress

Specifies the destination party to which the call leg should be routed.

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

originalDestinationAddress : in TpAddress

Specifies the original destination address of the call.

redirectingAddress : in TpAddress

Specifies the address from which the call was last redirected.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call (such as alerting method, tele-service type, service identities and interaction indicators).

callLegSessionID : out TpSessionIDRef

Specifies the sessionID assigned by the gateway. This is the sessionID of the implicitly created call leg. The same ID will be returned in the routeRes or Err. This allows the application to correlate the request and the result.

This parameter is only relevant when multiple routeReq() calls are executed in parallel, e.g., in the multi-party call control service.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_ADDRESS,P_UNSUPPORTED_ADDRESS_PLAN,P_INVALID_NETWORK_STATE,P_INVALID_CRITERIA,P_INVALID_EVENT_TYPE**

Method
release()

This method requests the release of the call object and associated objects. The call will also be terminated in the network. If the application requested reports to be sent at the end of the call (e.g., by means of getCallInfoReq) these reports will still be sent to the application.

The application should always either release or deassign the call when it is finished with the call, unless a callFaultDetected is received by the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE**

Method
deassignCall()

This method requests 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, call information reports or call Leg information reports requested, then these reports will be disabled and any related information discarded.

The application should always either release or deassign the call when it is finished with the call, unless callFaultDetected is received by the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

Method
getCallInfoReq()

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method must be invoked before the call is routed to a target address. Two types of reports can be requested; a final report or intermediate reports.

A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.

Intermediate reports are received when the destination leg or party terminates or when the call ends. In case the originating party is still available the application can still initiate a follow-on call using routeReq.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType

Specifies the call information that is requested.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

Method

setCallChargePlan()

Set an operator specific charge plan for the call. The charge plan must be set before the call is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tariffSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

~~**TpGeneralException,TpGCCSException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

Method

getMoreDialledDigitsReq()

This asynchronous method requests the call control service to collect further digits and return them to the application. Depending on the administered data, the network may indicate a new call to the gateway if a caller goes off-hook or dialled only a few digits. The application then gets a new call event which contains no digits or only the few dialled digits in the event data.

The application should use this method if it requires more dialled digits, e.g. to perform screening.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

length : in TpInt32

Specifies the maximum number of digits to collect.

Raises

~~**TpGeneralException, TpGCCSException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

Method

superviseCallReq()

The application calls this method to supervise a call. The application can set a granted connection time for this call. If an application calls this function before it calls a routeReq() or a user interaction function the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

Specifies how the network should react after the granted connection time expired.

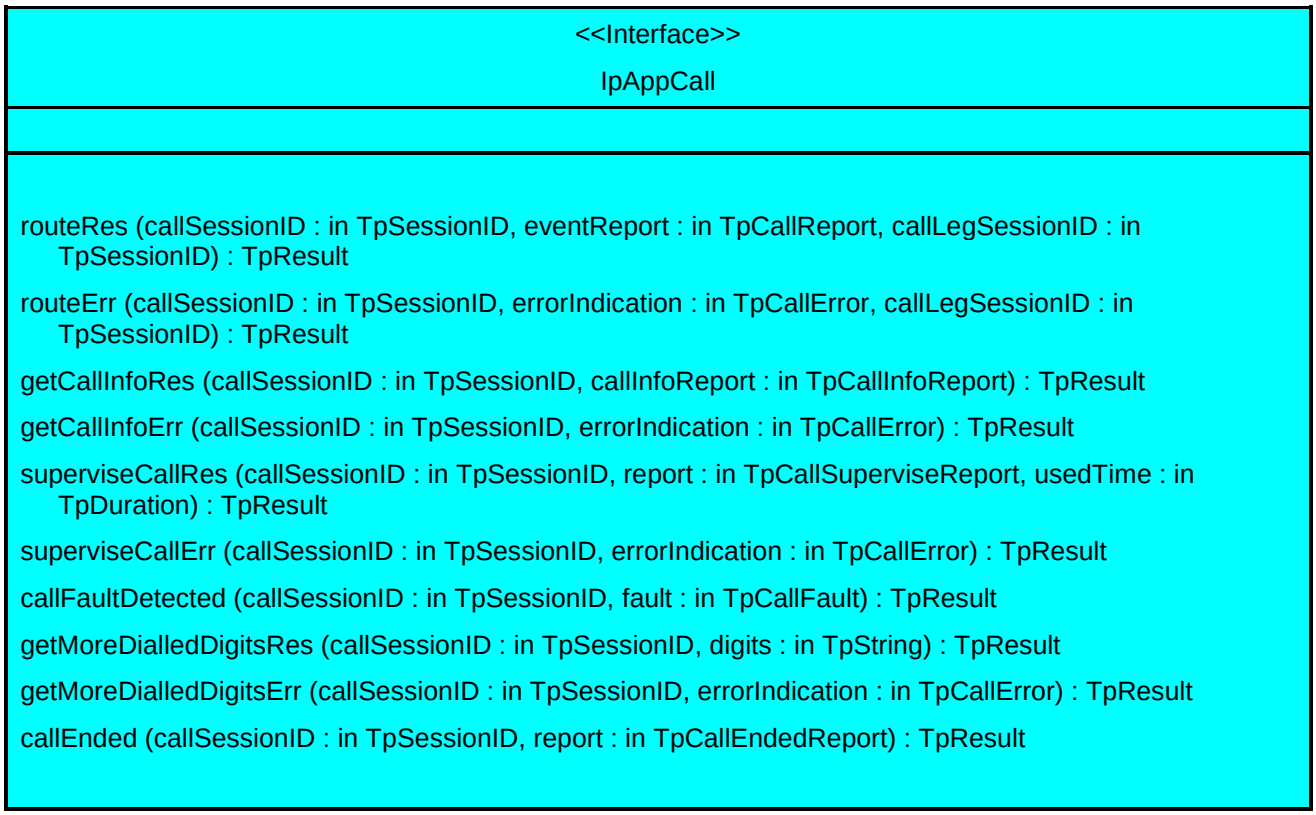
Raises

~~**TpGCCSException,TpGeneralException**~~**TpCommonExceptions,P_INVALID_SESSION_ID**

6.3.4 Interface Class IpAppCall

Inherits from: IpInterface

The generic call application interface is implemented by the client application developer and is used to handle call request responses and state reports.



Method

routeRes()

This asynchronous method indicates that the request to route the call to the destination was successful, and indicates the response of the destination party (for example, the call was answered, not answered, refused due to busy, etc.).

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED,

then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

eventReport : in TpCallReport

Specifies the result of the request to route the call to the destination party. It also includes the network event, date and time, monitoring mode and event specific information such as release cause.

callLegSessionID : in TpSessionID

Specifies the sessionID of the associated call leg. This corresponds to the sesion ID returned at the routeReq() and can be used to correlate the response with the request.

Raises

TpGCCSException,TpGeneralException

Method

routeErr()

This asynchronous method 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, the network was unable to route the call, the parameters were incorrect, the request was refused, etc.).

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

callLegSessionID : in TpSessionID

Specifies the sessionID of the associated call leg. This corresponds to the sessionID returned at the routeReq() and can be used to correlate the error with the request.

Raises

TpGCCSException,TpGeneralException

Method

getCallInfoRes()

This asynchronous method reports time information of the finished call or call attempt as well as release cause depending on which information has been requested by getCallInfoReq. This information may be used e.g. for charging purposes. The call information will possibly be sent after routeRes in all cases where the call or a leg of the call has been disconnected or a routing failure has been encountered.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoReport : in TpCallInfoReport

Specifies the call information requested.

Raises

TpGCCSException,TpGeneralException

Method

getCallInfoErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Raises

TpGCCSException,TpGeneralException

Method

superviseCallRes()

This asynchronous method reports a call supervision event to the application when it has indicated it's interest in these kind of events.

It is also called when the connection is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call

report : in TpCallSuperviseReport

Specifies the situation which triggered the sending of the call supervision response.

usedTime : in TpDuration

Specifies the used time for the call supervision (in milliseconds).

Raises

TpGCCSException,TpGeneralException

Method

superviseCallErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Raises

TpGCCSException,TpGeneralException

Method

callFaultDetected()

This method indicates to the application that a fault in the network has been detected. The call may or may not have been terminated.

The system deletes the call object. Therefore, the application has no further control of call processing. No report will be forwarded to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call in which the fault has been detected.

fault : in TpCallFault

Specifies the fault that has been detected.

Raises

TpGCCSException,TpGeneralException

Method

getMoreDialledDigitsRes()

This asynchronous method returns the collected digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

digits : in TpString

Specifies the additional dialled digits if the string length is greater than zero.

Raises

TpGeneralException,TpGCCSException

Method
getMoreDialledDigitsErr()

This asynchronous method reports an error in collecting digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Raises

TpGeneralException,TpGCCSException

Method
callEnded()

This method indicates to the application that the call has terminated in the network. However, the application may still receive some results (e.g., getCallInfoRes) related to the call. The application is expected to deassign the call object after having received the callEnded.

Note that the event that caused the call to end might also be received separately if the application was monitoring for it.

Parameters

callSessionID : in TpSessionID

Specifies the call sessionID.

report : in TpCallEndedReport

Specifies the reason the call is terminated.

Raises

TpGeneralException,TpGCCSException

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010704**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-04** **CR** **018** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to Call Control – Errors in Exceptions
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/08/2001
Category:	⌘	**F**
Release:	⌘	REL-4

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	- Consideration of exception handling in GCCS/MPCCS has identified further exceptions that are missing, and also erroneous exceptions
Summary of change:	⌘	Additional exceptions identified for following interfaces, IpCallControlManager, IpCall, IpMultiPartyCallControlManager, IpMultiPartyCall, IpCallLeg
Consequences if not approved:	⌘	Incomplete and inaccurate method specifications. Failure to adopt CR would result in divergence between 3GPP R4 specification and ETSI/Parlay specifications.

Clauses affected:	⌘	6.3 & 7.3
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	Clause 6.3 in this CR assumes that CR29.198-04-017_N5-010663 is accepted.

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

6.3 Generic Call Control Service Interface Classes

The Generic Call Control Service (GCCS) provides the basic call control service for the API. It is based around a third party model, which allows calls to be instantiated from the network and routed through the network.

The GCCS supports enough functionality to allow call routing and call management for today's Intelligent Network (IN) services in the case of a switched telephony network, or equivalent for packet based networks.

It is the intention of the GCCS that it could be readily specialised into call control specifications, for example, ITU-T recommendations H.323, ISUP, Q.931 and Q.2931, ATM Forum specification UNI3.1 and the IETF Session Initiation Protocol, or any other call control technology.

The adopted call model has the following objects. Note that not all of these concepts are used in the generic call.

* a call object. A call is a relation between a number of parties. The call object relates to the entire call view from the application. E.g., the entire call will be released when a release is called on the call. Note that different applications can have different views on the same physical call, e.g., one application for the originating side and another application for the terminating side. The applications will not be aware of each other, all 'communication' between the applications will be by means of network signalling. The API currently does not specify any feature interaction mechanisms.

* a call leg object. The leg object represents a logical association between a call and an address. The relationship includes at least the signalling relation with the party. The relation with the address is only made when the leg is routed. Before that the leg object is IDLE and not yet associated with the address.

* an address. The address logically represents a party in the call.

* a terminal. A terminal is the end-point of the signalling and/or media for a party. This object type is currently not addressed.

The call object is used to establish a relation between a number of parties by creating a leg for each party within the call.

Associated with the signalling relationship represented by the call leg, there may also be a bearer connection (e.g., in the traditional voice only networks) or a number (zero or more) of media channels (in multi-media networks).

A leg can be attached to the call or detached from the call. When the leg is attached, this means that media or bearer channels related to the legs are connected to the media or bearer channels of the other legs that are attached to the same call. I.e., only legs that are attached can 'speak' to each other. A leg can have a number of states, depending on the signalling received from or sent to the party associated with the leg. Usually there is a limit to the number of legs that are in being routed (i.e., the connection is being established) or connected to the call (i.e., the connection is established). Also, there usually is a limit to the number of legs that can be simultaneously attached to the same call.

Some networks distinguish between controlling and passive legs. By definition the call will be released when the controlling leg is released. All other legs are called passive legs. There can be at most one controlling leg per call. However, there is currently no way the application can influence whether a Leg is controlling or not.

There are two ways for an application to get the control of a call. The application can request to be notified of calls that meet certain criteria. When a call occurs in the network that meets these criteria, the application is notified and can control the call. Some legs will already be associated with the call in this case. Another way is to create a new call from the application.

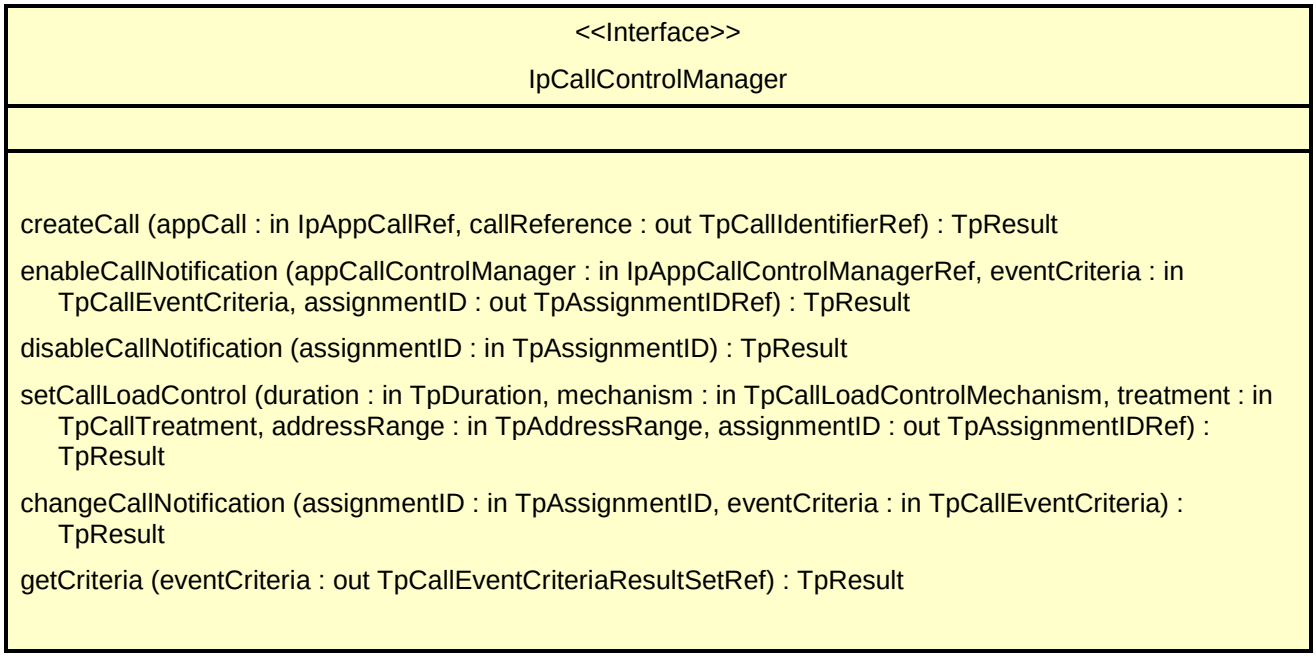
For the generic call control service, only a subset of the model is used; the API for generic call control does not give explicit access to the legs and the media channels. This is provided by the Multi-Party Call Control Service. Furthermore, the generic call is restricted to two party calls, i.e., only two legs are active at any given time. Active is defined here as 'being routed' or connected.

The GCCS is represented by the IpCallManager and IpCall interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppCallManager and IpAppCall to provide the callback mechanism.

6.3.1 Interface Class IpCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Generic Call Control Service. The generic call control manager interface provides the management functions to the generic call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications.



Method

createCall()

This method is used to create a new call object. An IpAppCallControlManager should already have been passed to the IpCallControlManager, otherwise the call control will not be able to report a callAborted()

to the application (the application should invoke setCallback() if it wishes to ensure this).

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

callReference : out TpCallIdentifierRef

Specifies the interface reference and sessionID of the call created.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

Method
enableCallNotification()

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notification of calls happening in the network. When such an event happens, the application will be informed by callEventNotify(). In case the application is interested in other events during the context of a particular call session it has to use the routeReq() method on the call object. The application will get access to the call object when it receives the callEventNotify(). (Note that the enableCallNotification() is not applicable if the call is setup by the application).

The enableCallNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P~~_GCCS~~_INVALID_CRITERIA.The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same CallNotificationType is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallBack().

Parameters

appCallControlManager : in IpAppCallControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

eventCriteria : in TpCallEventCriteria

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

assignmentID : out TpAssignmentIDRef

Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Raises

TpCommonExceptions,P_INVALID_CRITERIA,PINVALID_INTERFACE_TYPE,P_INVALID_EVENT_TYPE

Method
disableCallNotification()

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

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the generic call control manager interface when the previous enableNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENTID. If two callbacks have been registered under this assignment ID both of them will be disabled.

Raises

TpCommonExceptions,P_INVALID_ASSIGNMENT_ID

Method

setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the generic call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

Parameters

duration : in TpDuration

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

mechanism : in TpCallLoadControlMechanism

Specifies the load control mechanism to use (for example, admit one call per interval), and any necessary parameters, such as the call admission rate. The contents of this parameter are ignored if the load control duration is set to zero.

treatment : in TpCallTreatment

Specifies the treatment of calls that are not admitted. The contents of this parameter are ignored if the load control duration is set to zero.

addressRange : in TpAddressRange

Specifies the address or address range to which the overload control should be applied or removed.

assignmentID : out TpAssignmentIDRef

Specifies the assignmentID assigned by the gateway to this request. This assignementID can be used to correlate the callOverlloadEncountered and callOverloadCeased methods with the request.

Raises

TpCommonExceptions,P_INVALID_ADDRESS,P_UNSUPPORTED_ADDRESS_PLAN

Method

changeCallNotification()

This method is used by the application to change the event criteria introduced with enableCallNotification. Any stored criteria associated with the specified assignementID will be replaced with the specified criteria.

Parameters

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic call control manager interface for the event notification. If two call backs have been registered under this assignment ID both of them will be changed.

eventCriteria : in TpCallEventCriteria

Specifies the new set of event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

TpCommonExceptions,P_INVALID_ASSIGNMENT_ID,P_INVALID_CRITERIA,P_INVALID_EVENT_TYPE

Method

getCriteria()

This method is used by the application to query the event criteria set with enableCallNotification or changeCallNotification.

Parameters

eventCriteria : out TpCallEventCriteriaResultSetRef

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

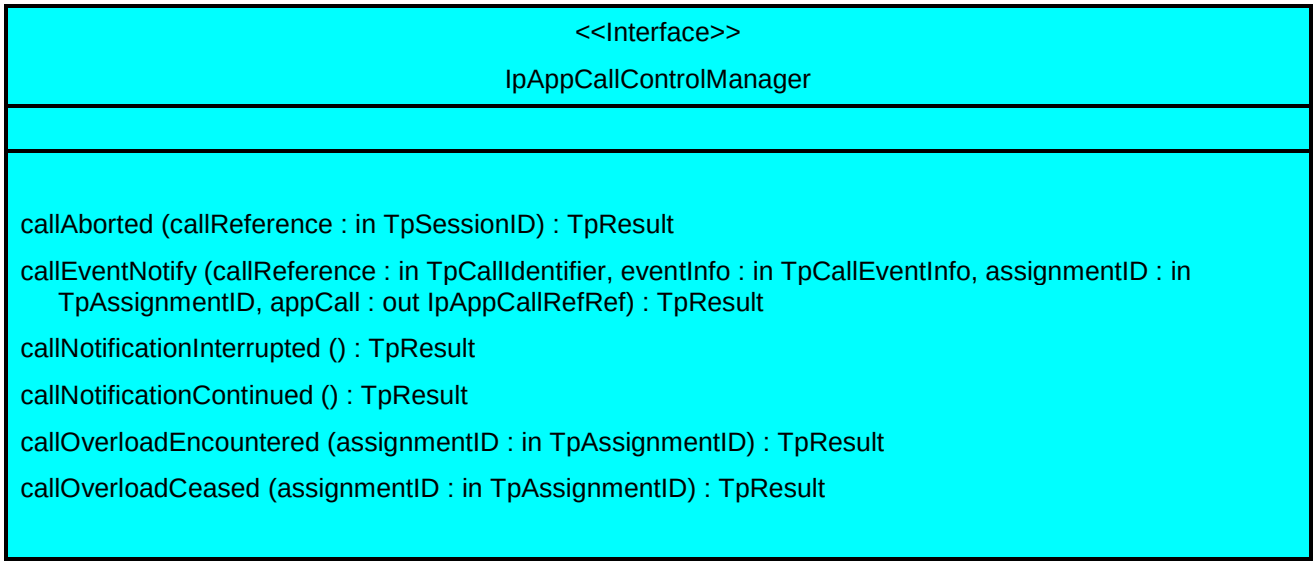
Raises

TpCommonExceptions

6.3.2 Interface Class IpAppCallControlManager

Inherits from: IpInterface

The generic call control manager application interface provides the application call control management functions to the generic call control service.



Method
callAborted()

This method indicates to the application that the call object (at the gateway) has aborted or terminated abnormally. No further communication will be possible between the call and application.

Parameters

callReference : in TpSessionID

Specifies the sessionID of call that has aborted or terminated abnormally.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method
callEventNotify()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

When this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPT, the application writer should ensure that no routeReq() is performed until an IpAppCall has been passed to the gateway, either through an explicit setCallback() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

Parameters

callReference : in TpCallIdentifier

Specifies the reference to the call interface to which the notification relates. This parameter will be null if the notification is in NOTIFY mode.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

Specifies the assignment id which was returned by the enableNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

appCall : out IpAppCallRefRef

Specifies a reference to the application interface which implements the callback interface for the new call. This parameter will be null if the notification is in NOTIFY mode.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method
callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporary interrupted (for example, due to faults detected).

Note that more permanent failures are reported via the Framework (integrity management).

Parameters
No Parameters were identified for this method

~~Raises~~

~~TpCommonExceptions~~

Method
callNotificationContinued()

This method indicates to the application that event notifications will again be possible.

Parameters
No Parameters were identified for this method

~~Raises~~

~~TpCommonExceptions~~

Method
callOverloadEncountered()

This method indicates 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.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been encountered.

~~Raises~~

~~TpCommonExceptions~~

Method
callOverloadCeased()

This method indicates 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.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been ceased

~~*Raises*~~

~~**TpCommonExceptions**~~

6.3.3 Interface Class IpCall

Inherits from: IpService

The generic Call provides the possibility to control the call routing, to request information from the call, control the charging of the call, to release the call and to supervise the call. It does not give the possibility to control the legs directly and it does not allow control over the media. The first capability is provided by the multi-party call and the latter as well by the multi-media call. The call is limited to two party calls, although it is possible to provide 'follow-on' calls, meaning that the call can be rerouted after the terminating party has disconnected or routing to the terminating party has failed. Basically, this means that at most two legs can be in connected or routing state at any time.

<<Interface>> IpCall
routeReq (callSessionID : in TpSessionID, responseRequested : in TpCallReportRequestSet, targetAddress : in TpAddress, originatingAddress : in TpAddress, originalDestinationAddress : in TpAddress, redirectingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, callLegSessionID : out TpSessionIDRef) : TpResult release (callSessionID : in TpSessionID, cause : in TpCallReleaseCause) : TpResult deassignCall (callSessionID : in TpSessionID) : TpResult getCallInfoReq (callSessionID : in TpSessionID, callInfoRequested : in TpCallInfoType) : TpResult setCallChargePlan (callSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : TpResult setAdviceOfCharge (callSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : TpResult getMoreDialledDigitsReq (callSessionID : in TpSessionID, length : in TpInt32) : TpResult superviseCallReq (callSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : TpResult

Method

routeReq()

This asynchronous method requests routing of the call to the remote party indicated by the targetAddress.

The extra address information such as originatingAddress is optional. If not present (i.e., the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in corresponding addresses from the route is used, otherwise the network or gateway provided numbers will be used.

If this method in invoked, and call reports have been requested, yet no IpAppCall interface has been provided, this method shall throw the P_NO_CALLBACK_ADDRESS_SET exception.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

responseRequested : in TpCallReportRequestSet

Specifies the set of observed events that will result in zero or more routeRes() being generated.

E.g., when both answer and disconnect is monitored the result can be received two times.

If the application wants to control the call (in whatever sense) it shall enable event reports

targetAddress : in TpAddress

Specifies the destination party to which the call leg should be routed.

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

originalDestinationAddress : in TpAddress

Specifies the original destination address of the call.

redirectingAddress : in TpAddress

Specifies the address from which the call was last redirected.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call (such as alerting method, tele-service type, service identities and interaction indicators).

callLegSessionID : out TpSessionIDRef

Specifies the sessionID assigned by the gateway. This is the sessionID of the implicitly created call leg. The same ID will be returned in the routeRes or Err. This allows the application to correlate the request and the result.

This parameter is only relevant when multiple routeReq() calls are executed in parallel, e.g., in the multi-party call control service.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_ADDRESS,P_UNSUPPORTED_ADDRESS_PLAN,P_INVALID_NETWORK_STATE,P_INVALID_CRITERIA,P_INVALID_EVENT_TYPE

Method

release()

This method requests the release of the call object and associated objects. The call will also be terminated in the network. If the application requested reports to be sent at the end of the call (e.g., by means of getCallInfoReq) these reports will still be sent to the application.

The application should always either release or deassign the call when it is finished with the call, unless a callFaultDetected is received by the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE

Method

deassignCall()

This method requests 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, call information reports or call Leg information reports requested, then these reports will be disabled and any related information discarded.

The application should always either release or deassign the call when it is finished with the call, unless callFaultDetected is received by the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID

Method

getCallInfoReq()

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method must be invoked before the call is routed to a target address. Two types of reports can be requested; a final report or intermediate reports.

A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.

Intermediate reports are received when the destination leg or party terminates or when the call ends. In case the originating party is still available the application can still initiate a follow-on call using routeReq.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType

Specifies the call information that is requested.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID

Method

setCallChargePlan()

Set an operator specific charge plan for the call. The charge plan must be set before the call is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID, P_INVALID_CURRENCY, P_INVALID_AMOUNT

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tariffSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID, P_INVALID_CURRENCY, P_INVALID_AMOUNT

Method

getMoreDialledDigitsReq()

This asynchronous method requests the call control service to collect further digits and return them to the application. Depending on the administered data, the network may indicate a new call to the gateway if a caller goes off-hook or dialled only a few digits. The application then gets a new call event which contains no digits or only the few dialled digits in the event data.

The application should use this method if it requires more dialled digits, e.g. to perform screening.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

length : in TpInt32

Specifies the maximum number of digits to collect.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID

Method

superviseCallReq()

The application calls this method to supervise a call. The application can set a granted connection time for this call. If an application calls this function before it calls a routeReq() or a user interaction function the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

Specifies how the network should react after the granted connection time expired.

Raises

TpCommonExceptions,P_INVALID_SESSION_ID

6.3.4 Interface Class IpAppCall

Inherits from: IpInterface

The generic call application interface is implemented by the client application developer and is used to handle call request responses and state reports.

<<Interface>> IpAppCall
routeRes (callSessionID : in TpSessionID, eventReport : in TpCallReport, callLegSessionID : in TpSessionID) : TpResult routeErr (callSessionID : in TpSessionID, errorIndication : in TpCallError, callLegSessionID : in TpSessionID) : TpResult getCallInfoRes (callSessionID : in TpSessionID, callInfoReport : in TpCallInfoReport) : TpResult getCallInfoErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult superviseCallRes (callSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : TpResult superviseCallErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult callFaultDetected (callSessionID : in TpSessionID, fault : in TpCallFault) : TpResult getMoreDialledDigitsRes (callSessionID : in TpSessionID, digits : in TpString) : TpResult getMoreDialledDigitsErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult callEnded (callSessionID : in TpSessionID, report : in TpCallEndedReport) : TpResult

Method

routeRes()

This asynchronous method indicates that the request to route the call to the destination was successful, and indicates the response of the destination party (for example, the call was answered, not answered, refused due to busy, etc.).

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED,

then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

eventReport : in TpCallReport

Specifies the result of the request to route the call to the destination party. It also includes the network event, date and time, monitoring mode and event specific information such as release cause.

callLegSessionID : in TpSessionID

Specifies the sessionID of the associated call leg. This corresponds to the sesion ID returned at the routeReq() and can be used to correlate the response with the request.

~~Raises~~

~~TpCommonExceptions~~

Method

routeErr()

This asynchronous method 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, the network was unable to route the call, the parameters were incorrect, the request was refused, etc.).

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

callLegSessionID : in TpSessionID

Specifies the sessionID of the associated call leg. This corresponds to the sessionID returned at the routeReq() and can be used to correlate the error with the request.

~~Raises~~

~~TpCommonExceptions~~

Method

getCallInfoRes()

This asynchronous method reports time information of the finished call or call attempt as well as release cause depending on which information has been requested by getCallInfoReq. This information may be used e.g. for charging purposes. The call information will possibly be sent after routeRes in all cases where the call or a leg of the call has been disconnected or a routing failure has been encountered.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoReport : in TpCallInfoReport

Specifies the call information requested.

~~Raises~~

~~TpCommonExceptions~~

Method
getCallInfoErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

~~Raises~~

~~TpCommonExceptions~~

Method
superviseCallRes()

This asynchronous method reports a call supervision event to the application when it has indicated it's interest in these kind of events.

It is also called when the connection is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call

report : in TpCallSuperviseReport

Specifies the situation which triggered the sending of the call supervision response.

usedTime : in TpDuration

Specifies the used time for the call supervision (in milliseconds).

~~Raises~~

~~TpCommonExceptions~~

Method
superviseCallErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method

callFaultDetected()

This method indicates to the application that a fault in the network has been detected. The call may or may not have been terminated.

The system deletes the call object. Therefore, the application has no further control of call processing. No report will be forwarded to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call in which the fault has been detected.

fault : in TpCallFault

Specifies the fault that has been detected.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method

getMoreDialledDigitsRes()

This asynchronous method returns the collected digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

digits : in TpString

Specifies the additional dialled digits if the string length is greater than zero.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method
getMoreDialledDigitsErr()

This asynchronous method reports an error in collecting digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

~~*Raises*~~

~~**TpCommonExceptions**~~

Method
callEnded()

This method indicates to the application that the call has terminated in the network. However, the application may still receive some results (e.g., getCallInfoRes) related to the call. The application is expected to deassign the call object after having received the callEnded.

Note that the event that caused the call to end might also be received separately if the application was monitoring for it.

Parameters

callSessionID : in TpSessionID

Specifies the call sessionID.

report : in TpCallEndedReport

Specifies the reason the call is terminated.

~~*Raises*~~

~~**TpCommonExceptions**~~

7.3 MultiParty Call Control Service Interface Classes

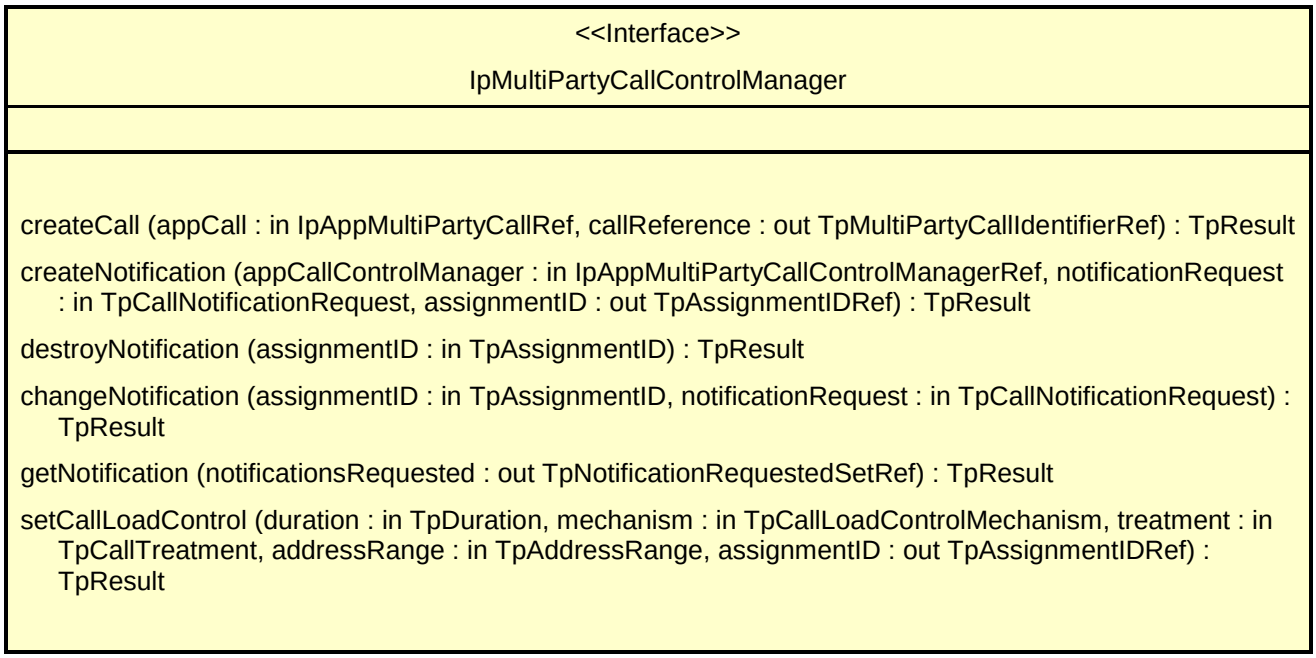
The Multi-party Call Control service enhances the functionality of the Generic Call Control Service with leg management. It also allows for multi-party calls to be established, i.e., up to a service specific number of legs can be connected simultaneously to the same call.

The Multi-party Call Control Service is represented by the IpMultiPartyCallControlManager, IpMultiPartyCall, IpCallLeg interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppMultiPartyCallControlManager, IpAppMultiPartyCall and IpAppCallLeg to provide the callback mechanism.

7.3.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Multi-party Call Control Service. The multi-party call control manager interface provides the management functions to the multi-party call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications. The action table associated with the STD shows in what state the IpMultiPartyCallControlManager must be if a method can successfully complete. In other words, if the IpMultiPartyCallControlManager is in another state the method will throw an exception immediately.



Method
`createCall()`

This method is used to create a new call object. An IpAppMultiPartyCallControlManager should already have been passed to the IpMultiPartyCallControlManager,

otherwise the call control will not be able to report a callAborted() to the application (the application should invoke setCallback() if it wishes to ensure this).

Parameters

appCall : in IpAppMultiPartyCallRef

Specifies the application interface for callbacks from the call created.

callReference : out TpMultiPartyCallIdentifierRef

Specifies the interface reference and sessionID of the call created.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

Method

createNotification()

This method is used to enable call notifications so that events can be sent to the application. This is 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(). In case the application is interested in other events during the context of a particular call session it has to use the createAndRouteCallLegReq() method on the call object or the eventReportReq() method on the call leg object. The application will get access to the call object when it receives thye reportNotification(). (Note that createNotification() is not applicable if the call is setup by the application).

The createNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P~~_GCCS~~_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same NotificationCallType is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as

callback the callback that has been registered by setCallback().

Parameters

appCallControlManager : in IpAppMultiPartyCallControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

notificationRequest : in TpCallNotificationRequest

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

assignmentID : out TpAssignmentIDRef

Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

Method

destroyNotification()

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

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the generic call control manager interface when the previous enableNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENTID. If two callbacks have been registered under this assignment ID both of them will be disabled.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

Method

changeNotification()

This method is used by the application to change the event criteria introduced with createNotification. Any stored criteria associated with the specified assignementID will be replaced with the specified criteria.

Parameters

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic call control manager interface for the event notification. If two callbacks have been registered under this assigment ID both of them will be disabled.

notificationRequest : in TpCallNotificationRequest

Specifies the new set of event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

Method

getNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Parameters

notificationsRequested : out TpNotificationRequestedSetRef

Specifies the nofications that have been requested by the application.

Raises

TpCommonExceptions

Method

setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

Parameters

duration : in TpDuration

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

mechanism : in TpCallLoadControlMechanism

Specifies the load control mechanism to use (for example, admit one call per interval), and any necessary parameters, such as the call admission rate. The contents of this parameter are ignored if the load control duration is set to zero.

treatment : in TpCallTreatment

Specifies the treatment of calls that are not admitted. The contents of this parameter are ignored if the load control duration is set to zero.

addressRange : in TpAddressRange

Specifies the address or address range to which the overload control should be applied or removed.

assignmentID : out TpAssignmentIDRef

Specifies the assignmentID assigned by the gateway to this request. This assignementID can be used to correlate the callOverlloadEncountered and callOverloadCeased methods with the request.

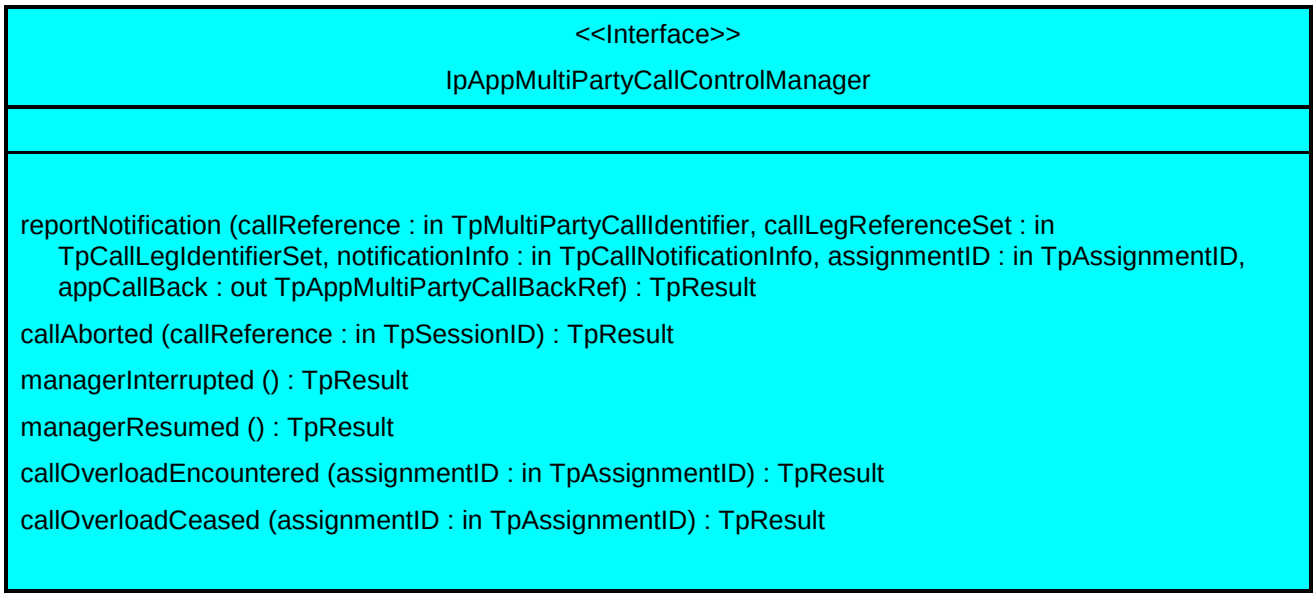
Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

7.3.2 Interface Class IpAppMultiPartyCallControlManager

Inherits from: IpInterface

The Multi-Party call control manager application interface provides the application call control management functions to the Multi-Party call control service.



Method

reportNotification()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of P_TIMER_EXPIRY.

Parameters

callReference : in TpMultiPartyCallIdentifier

Specifies the reference to the call interface to which the notification relates. This parameter will be null if the notification is being given in NOTIFY mode.

callLegReferenceSet : in TpCallLegIdentifierSet

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on who's behalf the notification was sent.

However, this parameter will be null if the notification is being given in NOTIFY mode.

notificationInfo : in TpCallNotificationInfo

Specifies data associated with this event (e.g. the originating or terminating leg which reports the notification).

assignmentID : in TpAssignmentID

Specifies the assignment id which was returned by the createNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

appCallBack : out TpAppMultiPartyCallBackRef

Specifies references to the application interface which implements the callback interface for the new call and/or new call leg. This parameter may be null if the notification is being given in NOTIFY mode.

Method
callAborted()

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

Parameters

callReference : in TpSessionID

Specifies the sessionID of call that has aborted or terminated abnormally.

Method
managerInterrupted()

This method indicates to the application that event notifications and method invocations have been temporary interrupted (for example, due to network resources unavailable).

Note that more permanent failures are reported via the Framework (integrity management).

Parameters

No Parameters were identified for this method

Method
managerResumed()

This method indicates to the application that event notifications possibleand method invocations are enabled.

Parameters

No Parameters were identified for this method

Method
callOverloadEncountered()

This method indicates 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.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been encountered.

Method
`callOverloadCeased()`

This method indicates 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.

Parameters

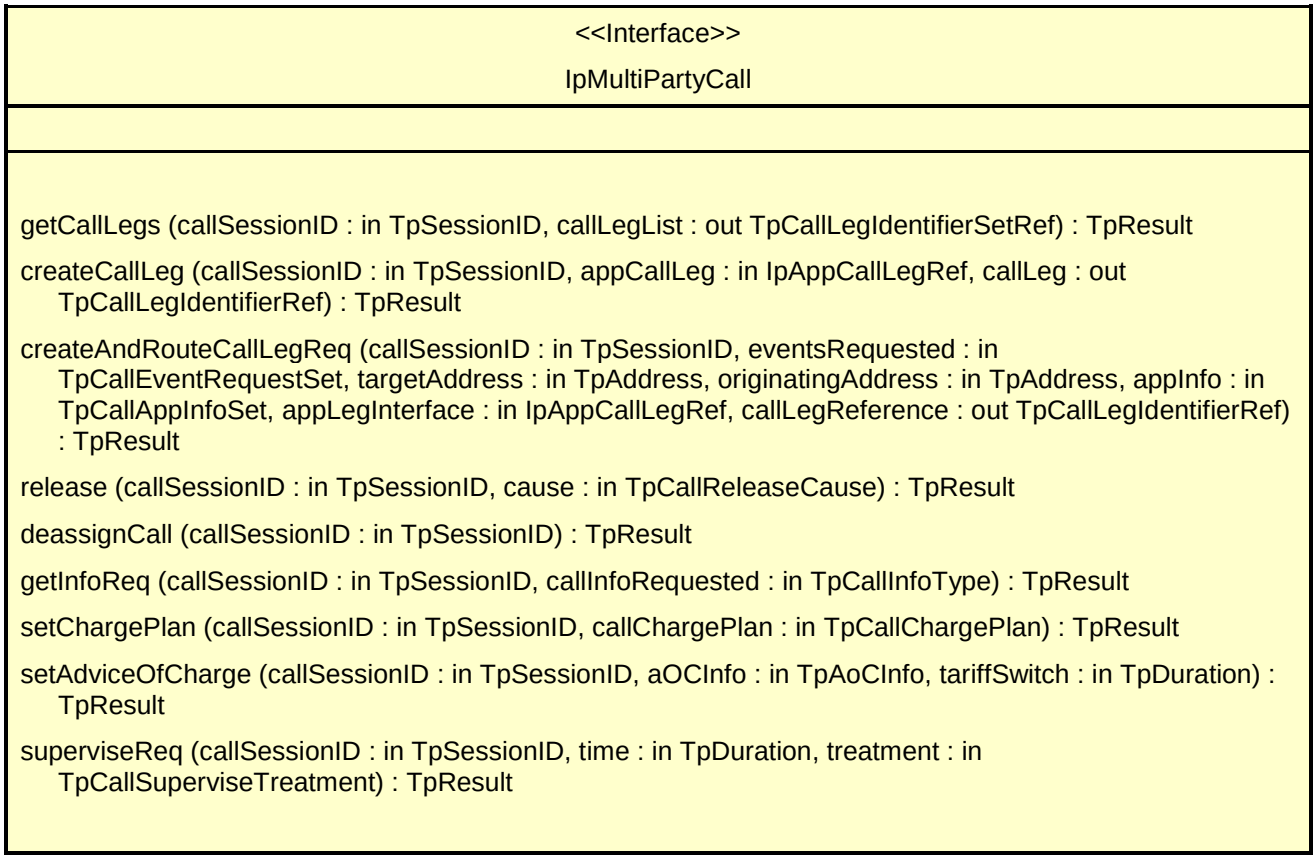
`assignmentID : in TpAssignmentID`

Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been ceased

7.3.3 Interface Class IpMultiPartyCall

Inherits from: IpService

The Multi-Party Call provides the possibility to control the call routing, to request information from the call, control the charging of the call, to release the call and to supervise the call. It also gives the possibility to manage call legs explicitly. An application may create more then one call leg.



Method
getCallLegs()

This method requests the identification of the call leg objects associated with the call object. Returns the legs in the order of creation.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callLegList : out TpCallLegIdentifierSetRef

Specifies the call legs associated with the call. The set contains both the sessionIDs and the interface references.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method
createCallLeg()

This method requests the creation of a new call leg object.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

appCallLeg : in IpAppCallLegRef

Specifies the application interface for callbacks from the call leg created.

callLeg : out TpCallLegIdentifierRef

Specifies the interface and sessionID of the call leg created.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

Method
createAndRouteCallLegReq()

This asynchronous operation requests creation and routing of a new callLeg. In case the connection to the destination party is established successfully the CallLeg is attached to the call, i.e. no explicit setMedia() operation is needed. Requested events will be reported on the IpAppCallLeg interface. This interface the application must provide through the appLegInterface parameter.

The extra address information such as originatingAddress is optional. If not present (i.e., the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in corresponding addresses from the route is used, otherwise the network or gateway provided numbers will be used.

If this method in invoked, and call reports have been requested, yet the IpAppCallLeg interface parameter is NULL, this method shall throw the P_NO_CALLBACK_ADDRESS_SET exception.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

eventsRequested : in TpCallEventRequestSet

Specifies the event specific criteria used by the application to define the events required. Only events that meet these criteria are reported. Examples of events are "adress analysed", "answer", "release".

targetAddress : in TpAddress

Specifies the destination party to which the call should be routed.

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call (such as alerting method, tele-service type, service identities and interaction indicators).

appLegInterface : in IpAppCallLegRef

Specifies a reference to the application interface that implements the callback interface for the new call leg. Requested events will be reported by the eventReportRes() operation on this interface.

callLegReference : out TpCallLegIdentifierRef

Specifies the reference to the CallLeg interface that was created.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_ADDRESS , P_UNSUPPORTED_ADDRESS_PLAN, P_INVALID_NETWORK_STATE, P_INVALID_EVENT_TYPE, P_INVALID_CRITERIA

Method

release()

This method requests the release of the call object and associated objects. The call will also be terminated in the network. If the application requested reports to be sent at the end of the call (e.g., by means of getInfoReq) these reports will still be sent to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

deassignCall()

This method requests 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 call information reports, call leg event reports or call Leg information reports requested, then these reports will be disabled and any related information discarded.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

getInfoReq()

This asynchronous method requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method must be invoked before the call is routed to a target address. Two types of reports can be requested; a final report or intermediate reports.

A final call report is sent when the call is ended. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The call information will be sent after any call event reports.

Intermediate reports are received when the destination leg or party terminates or when the call ends.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoRequested : in TpCallInfoType

Specifies the call information that is requested.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setChargePlan()

Set an operator specific charge plan for the call. The charge plan must be set before the call is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tariffSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_CURRENCY, P_INVALID_AMOUNT

Method

superviseReq()

The application calls this method to supervise a call. The application can set a granted connection time for this call. If an application calls this operation before it routes a call or a user interaction operation the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

Specifies how the network should react after the granted connection time expired.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

7.3.4 Interface Class IpAppMultiPartyCall

Inherits from: IpInterface

The Multi-Party call application interface is implemented by the client application developer and is used to handle call request responses and state reports.

<<Interface>> IpAppMultiPartyCall
getInfoRes (callSessionID : in TpSessionID, callInfoReport : in TpCallInfoReport) : TpResult getInfoErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult superviseRes (callSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : TpResult superviseErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult callFaultDetected (callSessionID : in TpSessionID, fault : in TpCallFault) : TpResult callEnded (callSessionID : in TpSessionID, report : in TpCallEndedReport) : TpResult createAndRouteCallLegErr (callSessionID : in TpSessionID, callLegReference : in TpCallLegIdentifier, errorIndication : in TpCallError) : TpResult

Method

getInfoRes()

This asynchronous method reports time information of the finished call or call attempt as well as release cause depending on which information has been requested by getInfoReq. This information may be used e.g. for charging purposes. The call information will possibly be sent after reporting of all cases where the call or a leg of the call has been disconnected or a routing failure has been encountered.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callInfoReport : in TpCallInfoReport

Specifies the call information requested.

Method
getInfoErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method
superviseRes()

This asynchronous method reports a call supervision event to the application when it has indicated it's interest in these kind of events.

It is also called when the connection is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call

report : in TpCallSuperviseReport

Specifies the situation which triggered the sending of the call supervision response.

usedTime : in TpDuration

Specifies the used time for the call supervision (in milliseconds).

Method
superviseErr()

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

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method
callFaultDetected()

This method indicates to the application that a fault in the network has been detected. The call may or may not have been terminated.

The system deletes the call object. Therefore, the application has no further control of call processing.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call in which the fault has been detected.

fault : in TpCallFault

Specifies the fault that has been detected.

Method
callEnded()

This method indicates to the application that the call has terminated in the network.

Note that the event that caused the call to end might have been received separately if the application was monitoring for it.

Parameters

callSessionID : in TpSessionID

Specifies the call sessionID.

report : in TpCallEndedReport

Specifies the reason the call is terminated.

Method
createAndRouteCallLegErr()

This asynchronous method 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, the network was unable to route the call, the parameters were incorrect, the request was refused, etc.). Note that the event cases that can be monitored and correspond to an unsuccessful setup of a connection (e.g. busy, no_answer) will be reported by eventReportRes() and not by this operation.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

callLegReference : in TpCallLegIdentifier

Specifies the reference to the CallLeg interface that was created.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

7.3.5 Interface Class IpCallLeg

Inherits from: The call leg interface represents the logical call leg associating a call with an address. The call leg tracks its own states and allows charging summaries to be accessed. The leg represents the signalling relationship between the call and an address. An application that uses the IpCallLeg interface to set up connections has more control, e.g. by defining leg specific event request and can obtain call leg specific report and events.

<<Interface>> IpCallLeg
routeReq (callLegSessionID : in TpSessionID, targetAddess : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, connectionProperties : in TpCallLegConnectionProperties) : TpResult eventReportReq (callLegSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet) : TpResult release (callLegSessionID : in TpSessionID, cause : in TpCallReleaseCause) : TpResult getInfoReq (callLegSessionID : in TpSessionID, callLegInfoRequested : in TpCallLegInfoType) : TpResult getCall (callLegSessionID : in TpSessionID, callReference : out TpMultiPartyCallIdentifierRef) : TpResult attachMedia (callLegSessionID : in TpSessionID) : TpResult detachMedia (callLegSessionID : in TpSessionID) : TpResult getLastRedirectedAddress (callLegSessionID : in TpSessionID, redirectedAddress : out TpAddressRef) : TpResult continueProcessing (callLegSessionID : in TpSessionID) : TpResult getMoreDialledDigitsReq (callLegSessionID : in TpSessionID, length : in TpInt32) : TpResult setChargePlan (callLegSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : TpResult setAdviceOfCharge (callLegSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tarrifSwitch : in TpDuration) : TpResult superviseReq (callLegSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallSuperviseTreatment) : TpResult deassign (callLegSessionID : in TpSessionID) : TpResult

Method

routeReq()

This asynchronous method requests routing of the call leg to the remote party indicated by the targetAddress.

In case the connection to the destination party is established successfully the CallLeg will be either detached or attached to the call based on the attach Mechanism values specified in the connectionProperties parameter.

The extra address information such as originatingAddress is optional. If not present (i.e. the plan is set to P_ADDRESS_PLAN_NOT_PRESENT), the information provided in the corresponding addresses from the route is used, otherwise network or gateway provided addresses will be used.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

targetAddess : in TpAddress

Specifies the destination party to which the call leg should be routed

originatingAddress : in TpAddress

Specifies the address of the originating (calling) party.

appInfo : in TpCallAppInfoSet

Specifies application-related information pertinent to the call leg (such as alerting method, tele-service type, service identities and interaction indicators).

connectionProperties : in TpCallLegConnectionProperties

Specifies the properties of the connection.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

Method

eventReportReq()

This asynchronous method sets, clears or changes the criteria for the events that the call leg object will be set to observe.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

eventsRequested : in TpCallEventRequestSet

Specifies the event specific criteria used by the application to define the events required. Only events that meet these criteria are reported. Examples of events are "address analysed", "answer", "release".

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_EVENT_TYPE, P_INVALID_CRITERIA

Method
release()

This method requests the release of the call leg. If successful, the associated address (party) will be released from the call, and the call leg deleted. Note that in some cases releasing the party may lead to release of the complete call in the network. The application will be informed of this with callEnded().

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpCallReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method
getInfoReq()

This asynchronous method requests information associated with the call leg to be provided at the appropriate time (for example, to calculate charging). Note: in the call leg information must be accessible before the objects of concern are deleted.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

callLegInfoRequested : in TpCallLegInfoType

Specifies the call leg information that is requested.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method
getCall()

This method requests the call associated with this call leg.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

callReference : out TpMultiPartyCallIdentifierRef

Specifies the interface and sessionID of the call associated with this call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

attachMedia()

This method requests that the call leg be attached to its call object. This will allow transmission on all associated bearer connections or media channels to and from other parties in the call. The call leg must be in the connected state for this method to complete successfully.

Parameters

callLegSessionID : in TpSessionID

Specifies the sessionID of the call leg to attach to the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

detachMedia()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the sessionID of the call leg to detach from the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

getLastRedirectedAddress()

Queries the last address the leg has been redirected to.

Parameters

callLegSessionID : in TpSessionID

Specifies the call session ID of the call leg.

redirectedAddress : out TpAddressRef

Specifies the last address where the call leg was redirected to.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

continueProcessing()

This operation continues processing of the call leg. Applications can invoke this operation after call leg processing was interrupted due to detection of a notification or event the application subscribed it's interest in.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

Method

getMoreDialledDigitsReq()

This asynchronous method requests to collect further digits and return them to the application. Depending on the administered data, the network may indicate a new call to the gateway if a caller goes off-hook or dialled only a few digits. The application then gets a new call event which contains no digits or only the few dialled digits in the event data. The application should then use this method if it requires more dialled digits, e.g. to perform screening.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call.

length : in TpInt32

Specifies the maximum number of digits to collect.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setChargePlan()

Set an operator specific charge plan for the cal leg. The charge plan must be set before the call leg is routed to a target address. Depending on the operator the method can also be used to change the charge plan for ongoing calls.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

callChargePlan : in TpCallChargePlan

Specifies the charge plan to use.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setAdviceOfCharge()

This method allows for advice of charge (AOC) information to be sent to terminals that are capable of receiving this information.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

aOCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter.

tarrifSwitch : in TpDuration

Specifies the tariff switch interval that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_CURRENCY, P_INVALID_AMOUNT

Method

superviseReq()

The application calls this method to supervise a call leg. The application can set a granted connection time for this call. If an application calls this function before it calls a routeReq() or a user interaction function the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call party.

time : in TpDuration

Specifies the granted time in milliseconds for the connection.

treatment : in TpCallSuperviseTreatment

Specifies how the network should react after the granted connection time expired.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

deassign()

This method requests that the relationship between the application and the call leg and associated objects be de-assigned. It leaves the call leg 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 leg information reports requested, then these reports will be disabled and any related information discarded.

The application should always either release or deassign the call leg when it is finished with the call, leg unless callFaultDetected is received by the application.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

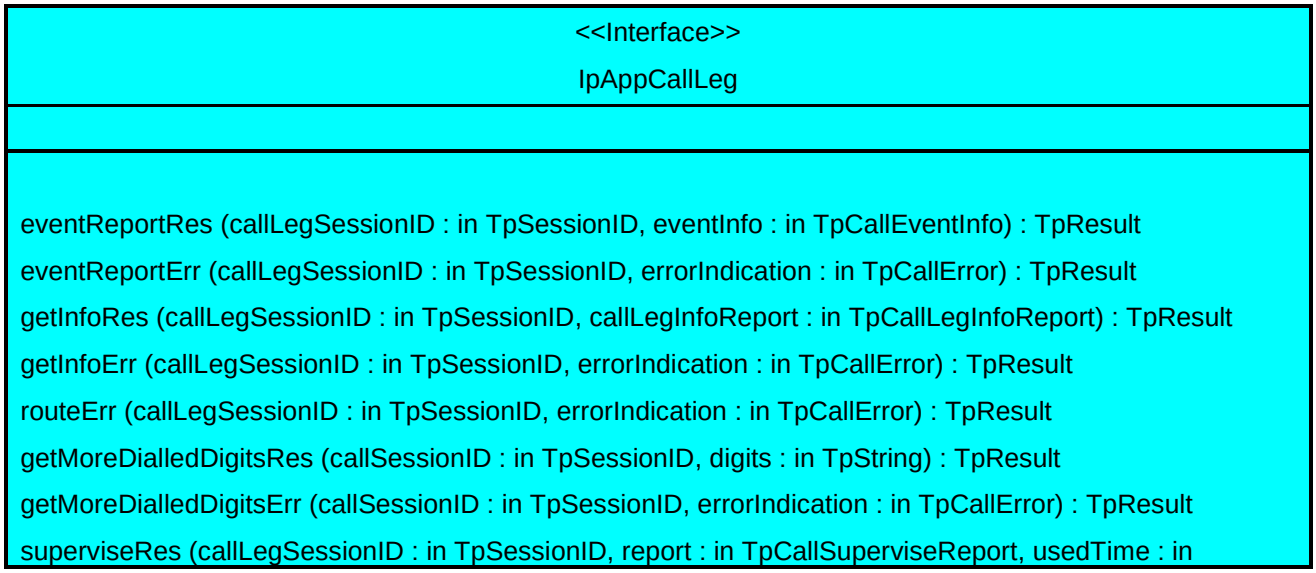
TpCommonExceptions, P_INVALID_SESSION_ID

7.3.6 Interface Class IpAppCallLeg

Inherits from: IpInterface

IpService

The application call leg interface is implemented by the client application developer and is used to handle responses and errors associated with requests on the call leg in order to be able to receive leg specific information and events.



```
    TpDuration) : TpResult
superviseErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : TpResult
connectionEnded (callLegSessionID : in TpSessionID, cause : in TpCallReleaseCause) : TpResult
```

Method

eventReportRes()

This asynchronous method reports that an event has occurred that was requested to be reported (for example, a mid-call event, the party has requested to disconnect, etc.).

Depending on the type of event received, outstanding requests for events are discarded. The exact details of these so-called disarming rules are captured in the data definition of

the event type.

If this method is invoked with a monitor mode of P_MONITOR_MODE_INTERRUPTED, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of P_TIMER_EXPIRY.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg on which the event was detected.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

Method

eventReportErr()

This asynchronous method indicates that the request to manage call leg event reports was unsuccessful, and the reason (for example, the parameters were incorrect, the request was refused, etc.).

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method

getInfoRes()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg to which the information relates.

callLegInfoReport : in TpCallLegInfoReport

Specifies the call leg information requested.

Method

getInfoErr()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method

routeErr()

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method

getMoreDialledDigitsRes()

This asynchronous method returns the collected digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

digits : in TpString

Specifies the additional dialled digits if the string length is greater than zero.

Method

getMoreDialledDigitsErr()

This asynchronous method reports an error in collecting digits to the application.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method

superviseRes()

This asynchronous method reports a call leg supervision event to the application when it has indicated its interest in these kind of events.

It is also called when the connection to a party is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg

report : in TpCallSuperviseReport

Specifies the situation which triggered the sending of the call leg supervision response.

usedTime : in TpDuration

Specifies the used time for the call leg supervision (in milliseconds).

Method

superviseErr()

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

errorIndication : in TpCallError

Specifies the error which led to the original request failing.

Method

connectionEnded()

This method indicates to the application that the connection has terminated in the network. However, the application may still receive some results (e.g., getInfoRes) related to the call leg. The application is expected to deassign the call leg object after having received the connectionEnded.

Note that the event that caused the connection to end might also be received separately if the application was monitoring for it.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpCallReleaseCause

Specifies the reason the connection is terminated.