3GPP TSG CN Plenary Meeting #19 **NP-030020**
12- 14 March 2003, Birmingham, UK

Source: **CN5 (OSA)**

Title: **Rel-4 CRs 29.198-04 OSA API Part 4: Call control**

Agenda item: **7.10**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
NP-030020	29.198-04	058	-	Rel-4	Correction of status of methods to interfaces in clause 6.3	F	4.5.0	N5-020888	OSA1
NP-030020	29.198-04-2	003	-	Rel-5	Correction of status of GCC methods	A	5.1.0	N5-020873	OSA2
NP-030020	29.198-04	059	-	Rel-4	Correction to TpReleaseCauseSet in Multi Party Call Control	F	4.5.0	N5-021052	OSA1
NP-030020	29.198-04-3	009	-	Rel-5	Correction to TpReleaseCauseSet in Multi Party Call Control IDL	A	5.1.0	N5-021053	OSA2
NP-030020	29.198-04	060	-	Rel-4	Correction to Sequence Diagrams to remove incorrect Framework references	F	4.5.0	N5-021063	OSA1
NP-030020	29.198-04	061	-	Rel-4	Correction to User Interaction Prepaid Sequence Diagrams	F	4.5.0	N5-021064	OSA1
NP-030020	29.198-04-2	004	-	Rel-5	Correction to Prepaid Sequence Diagram	A	5.1.0	N5-021065	OSA2
NP-030020	29.198-04	062	-	Rel-4	Correction to remove unused TpCallChargeOrder	F	4.5.0	N5-021079	OSA1
NP-030020	29.198-04-1	004	-	Rel-5	Correction to remove unused TpCallChargeOrder	A	5.1.0	N5-021080	OSA2
NP-030020	29.198-04	063	-	Rel-4	Correction to TpCallEventCriteriaResult in Generic Call Control	F	4.5.0	N5-021121	OSA1
NP-030020	29.198-04-2	005	-	Rel-5	Correction to TpCallEventCriteriaResult in Generic Call Control	A	5.1.0	N5-021122	OSA2
NP-030020	29.198-04	064	-	Rel-4	Correction of status of methods to interfaces in clause 7.3	F	4.5.0	N5-030048	OSA1

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-1** **CR** **004** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction to remove unused TpCallChargeOrder
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	31/10/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	After the charging mechanism was re-worked for Release 4 / Parlay 3.0 in the San Diego meeting, TpCallChargeOrder was no longer used. But it was not removed from the specification. Also TpCallChargePlan has an error in the description of its ChargePlan element.
Summary of change:	⌘	Remove the TpCallChargeOrder type (this is backwards compatible) Correct the description associated with the ChargePlan element of TpCallChargePlan
Consequences if not approved:	⌘	The description of TpCallChargePlan is very misleading and will confuse developers. Leaving unused types in the specification may confuse developers, who may not be able to see as easily as we can that the type is unused. This is especially true of the charging data types.

Clauses affected:	⌘	6.3, 6.6			
Other specs affected:	⌘	**Y**	**N**		
			X	Other core specifications	⌘
			X	Test specifications	
			X	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/specs/CR.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 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
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 change plan ~~transparent charging~~ is selected.
AdditionalInfo	TpOctetSet	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
PartyToCharge	TpCallPartyToChargeType	Identifies the entity or party to be charged for the call or call leg.
PartyToChargeAdditionalInfo	TpCallPartyToChargeAdditionalInfo	Contains additional information regarding the charged party.

6.4 TpCallPartyToChargeAdditionalInfo

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~~,~~	NULL	Undefined
P_CALL_PARTY_SPECIAL	TpAddress	CallPartySpecial

6.5 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

~~6.6TpCallChargeOrder~~

~~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_TRANSPARENT~~	~~TpOctetSet~~	~~TransparentCharge~~
~~P_CALL_CHARGE_PREDEFINED_SET~~	~~TpInt32~~	~~ChargePlan~~

~~6.7~~6.6 TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
P_CALL_CHARGE_TRANSPARENT	0	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	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).

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04 CR 064 ⌘rev - ⌘ Current version: 4.5.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title: ⌘	Correction of status of methods to interfaces in clause 7.3	
Source: ⌘	N5	
Work item code: ⌘	OSA1	**Date:** ⌘ 31/01/2003
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)*
Rel-6	*(Release 6)*

Reason for change: ⌘	There is no requirement in the standard about the necessity to implement all or only some of the methods defined for an interface.
Summary of change: ⌘	Clarify which methods are mandatory and which are optional.
Consequences if not approved: ⌘	Application developers will not know which methods will actually be available.

Clauses affected: ⌘	7.3 Multi Party Call Control Service Interface Classes			
Other specs affected: ⌘	**Y**	**N**		⌘
		X	Other core specifications	
		X	Test specifications	
		X	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/specs/CR.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 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.

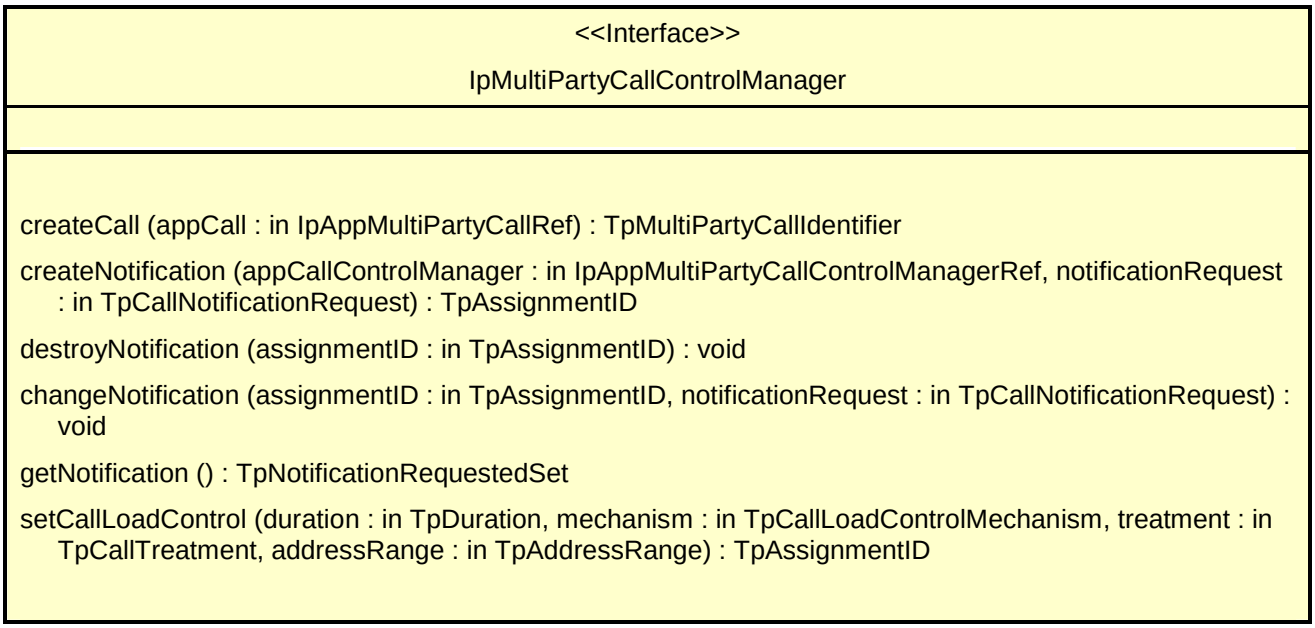
The Multi-party Call Control Service is represented by the IpMultiPartyCallControlManager, IpMultiPartyCall, IpCallLeg interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, 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.

This interface shall be implemented by a Multi Party Call Control SCF. As a minimum requirement either the createCall() method shall be implemented, or the createNotification() and destroyNotification() methods shall be implemented.



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).

Returns callReference: Specifies the interface reference and sessionID of the call created.

Parameters

appCall : in IpAppMultiPartyCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpMultiPartyCallIdentifier

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 the 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_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan 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().

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

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.

Returns

TpAssignmentID

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 exception P_INVALID_ASSIGNMENTID will be raised. 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 assignmentID 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 assignment ID both of them will be changed.

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.

Returns notificationsRequested: Specifies the notifications that have been requested by the application.

Parameters
No Parameters were identified for this method

Returns

TpNotificationRequestedSet

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.

Returns assignmentID: Specifies the assignmentID assigned by the gateway to this request. This assignmentID can be used to correlate the callOverloadEncountered and callOverloadCeased methods with the request.

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.

Returns

TpAssignmentID

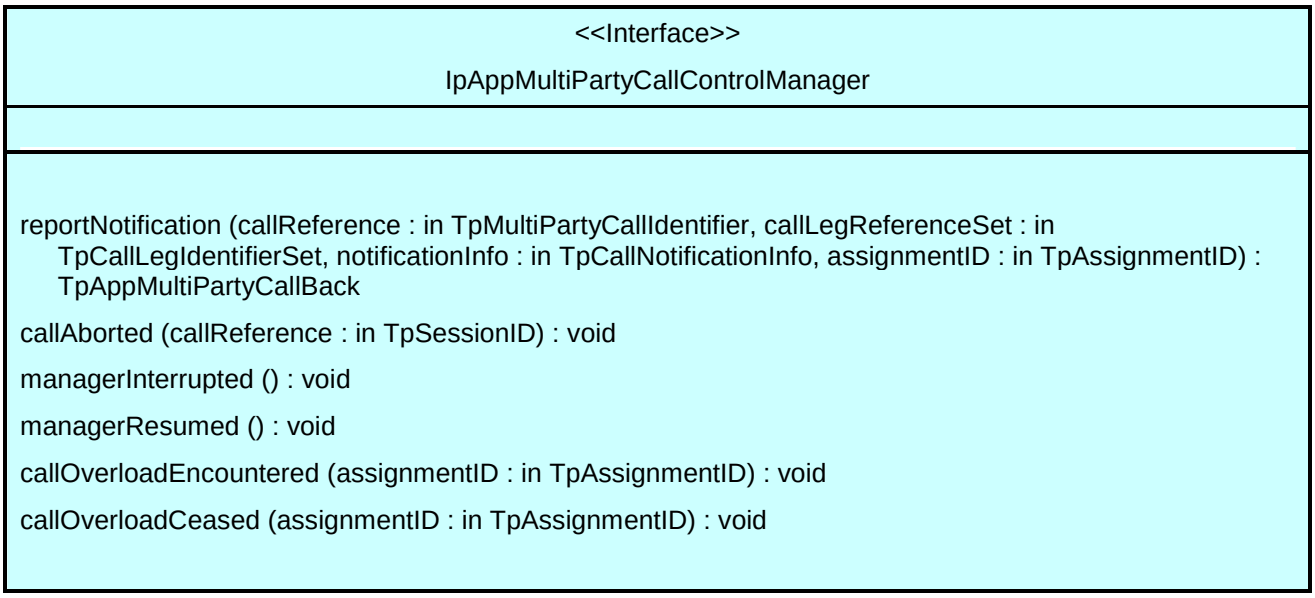
Raises

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_CALL_MONITOR_MODE_INTERRUPT, 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.

Returns appCallBack: Specifies references to the application interface which implements the callback interface for the new call and/or new call leg. If the application has previously explicitly passed a reference to the callback interface using a setCallback() invocation, this parameter may be set to P_APP_CALLBACK_UNDEFINED, or if supplied must be the same as that provided during the setCallback().

This parameter will be set to P_APP_CALLBACK_UNDEFINED if the notification is in NOTIFY mode.

Parameters

callReference : in TpMultiPartyCallIdentifier

Specifies the reference to the call interface to which the notification relates. If the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

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 whose behalf the notification was sent.

However, if the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

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.

Returns

TpAppMultiPartyCallBack

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 temporarily 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 are possible and 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.

This interface shall be implemented by a Multi Party Call Control SCF. The release() and deassignCall() methods, and either the createCallLeg() or the createAndRouteCallLegReq(), shall be implemented as a minimum requirement.

<<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 TpReleaseCause) : 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

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 attachMediaReq() 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 is 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 "address analysed", "answer" and "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_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 TpReleaseCause

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.

A report 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

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setChargePlan()

Set an operator specific charge plan for the call.

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) : 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 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 its 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
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 leg to the destination party was unsuccessful - the call leg could not be routed to the destination party (for example, the network was unable to route the call leg, 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

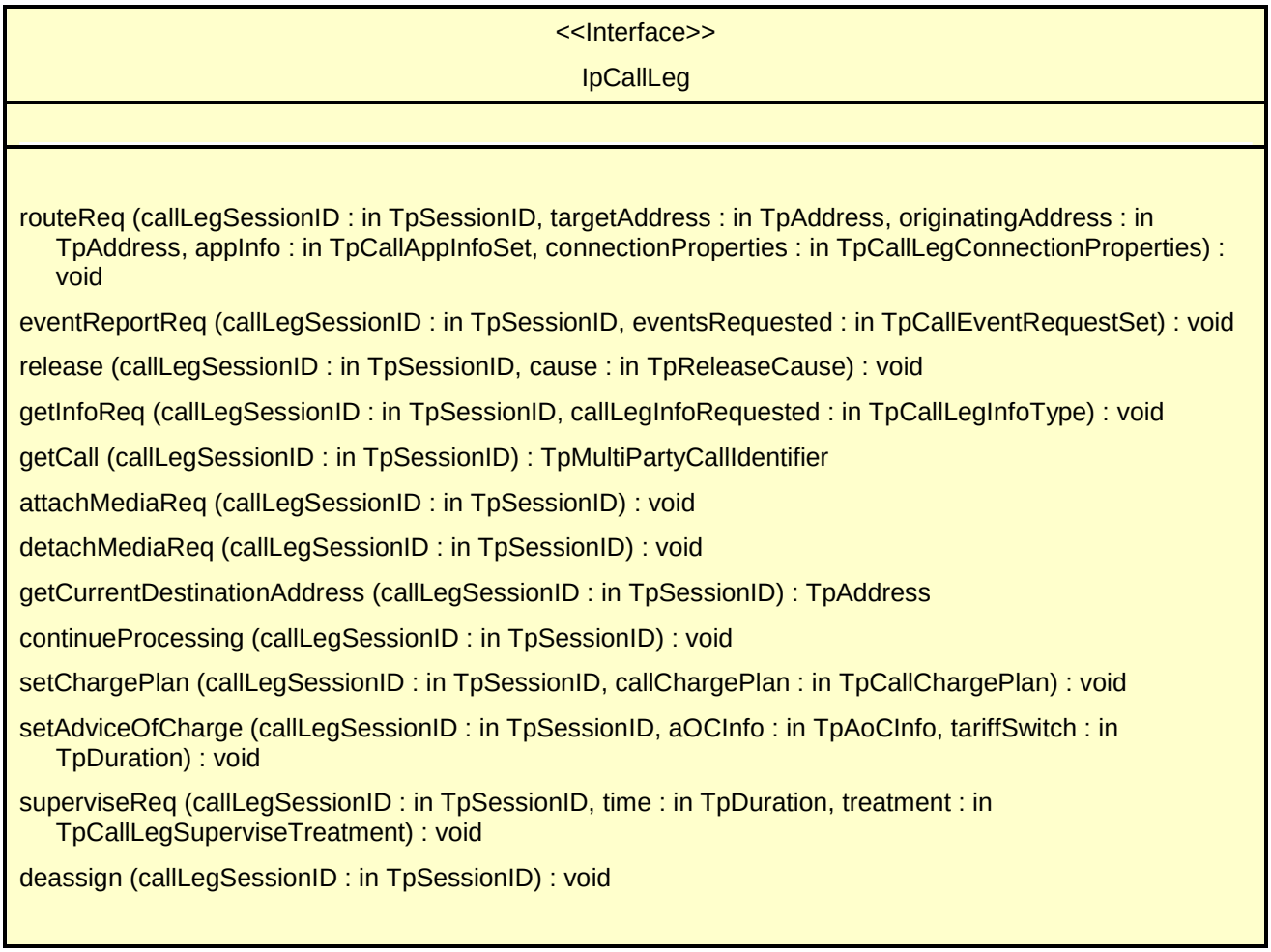
Specifies the error which led to the original request failing.

7.3.5 Interface Class IpCallLeg

Inherits from: IpService

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 control, e.g. by defining leg specific event request and can obtain call leg specific report and events.

This interface shall be implemented by a Multi Party Call Control SCF. The routeReq(), eventReportReq(), release(), continueProcessing() and deassign() methods shall be implemented as a minimum requirement.



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.

This operation continues processing of the call leg.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

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.

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" and "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().

This operation continues processing of the call leg.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpReleaseCause

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 that 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

attachMediaReq()

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 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

detachMediaReq()

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

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

getCurrentDestinationAddress()

Queries the current address of the destination the leg has been directed to.

Returns the address of the destination point towards which the call leg has been routed..

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.

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 its 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

setChargePlan()

Set an operator specific charge plan for the call leg.

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.

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 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 TpCallLegSuperviseTreatment

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 not release or deassign the call leg when received a callLegEnded() or callEnded(). 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

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) : void eventReportErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void attachMediaRes (callLegSessionID : in TpSessionID) : void attachMediaErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void detachMediaRes (callLegSessionID : in TpSessionID) : void detachMediaErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void getInfoRes (callLegSessionID : in TpSessionID, callLegInfoReport : in TpCallLegInfoReport) : void getInfoErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void routeErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void superviseRes (callLegSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : void superviseErr (callLegSessionID : in TpSessionID, errorIndication : in TpCallError) : void callLegEnded (callLegSessionID : in TpSessionID, cause : in TpReleaseCause) : void

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_CALL_MONITOR_MODE_INTERRUPT, then the application has control of the call leg. If the application does nothing with the call leg within a specified time period (the duration which forms a part of the service level agreement), then the 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

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

attachMediaRes()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg to which the information relates.

Method

attachMediaErr()

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

detachMediaRes()

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

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg to which the information relates.

Method

detachMediaErr()

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

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()

This asynchronous method indicates that the request to route the call leg to the destination party was unsuccessful - the call leg could not be routed to the destination party (for example, the network was unable to route the call leg, 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
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
callLegEnded()

This method indicates to the application that the leg has terminated in the network. The application has received all requested results (e.g., getInfoRes) related to the call leg. The call leg will be destroyed after returning from this method.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

cause : in TpReleaseCause

Specifies the reason the connection is terminated.

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04 CR 063 ⌘rev - ⌘ Current version: 4.5.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title:	⌘	Correction to TpCallEventCriteriaResult in Generic Call Control
Source:	⌘	N5
Work item code:	⌘	OSA1 **Date:** ⌘ 31/10/2002
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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	The text description for TpCallEventCriteriaResult contains a name of an element in a struct data type which contradicts the IDL description of the same data type in gcc_data.idl:
Summary of change:	⌘	Change text description to match current IDL description of TpCallEventCriteriaResult
Consequences if not approved:	⌘	A contradiction will exist between the IDL and the Word document. If no alignment is made, some developers will chose one name, others the other, and interworking problems will arise.

Clauses affected:	⌘	6.6.2.25

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	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/specs/CR.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.

6.6.2.25 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
CallEventCriteria	TpCallEventCriteria	The event criteria that were specified by the application.
AssignmentID	TpInt32	The associated assignmentID. This can be used to disable the notification.

←========== MODIFIED SECTION ==========→

Annex A (normative): OMG IDL Description of Call Control SCF

The OMG IDL representation of this interface specification is contained in text files (contained in archive 2919804IDL.ZIP) which accompany the present document.

In file gcc_data.idl:

```
struct TpCallEventCriteriaResult {
    TpCallEventCriteria CallEventCriteria;
    TpInt32 AssignmentID;
};
```

←========== END MODIFIED SECTION ==========→

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04** **CR** **062** ⌘ **rev** **-** ⌘ Current version: **4.5.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction to remove unused TpCallChargeOrder
Source:	⌘	N5
Work item code:	⌘	OSA1
Date:	⌘	31/10/2002
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)
Rel-6 (Release 6)

Reason for change:	⌘	After the charging mechanism was re-worked for Release 4 / Parlay 3.0 in the San Diego meeting, TpCallChargeOrder was no longer used. But it was not removed from the specification. Also TpCallChargePlan has an error in the description of its ChargePlan element.
Summary of change:	⌘	Remove the TpCallChargeOrder type (this is backwards compatible) Correct the description associated with the ChargePlan element of TpCallChargePlan
Consequences if not approved:	⌘	The description of TpCallChargePlan is very misleading and will confuse developers. Leaving unused types in the specification may confuse developers, who may not be able to see as easily as we can that the type is unused. This is especially true of the charging data types.

Clauses affected:	⌘	8.3, 8.6			
Other specs affected:	⌘	**Y**	**N**		
			X	Other core specifications	⌘
			X	Test specifications	
			X	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/specs/CR.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.

8.3 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
`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~~predefined change plan is selected.
`AdditionalInfo`	`TpOctetSet`	Descriptive string which is sent to the billing system without prior evaluation. Could be included in the ticket.
`PartyToCharge`	`TpCallPartyToChargeType`	Identifies the entity or party to be charged for the call or call leg.
`PartyToChargeAdditionalInfo`	`TpCallPartyToChargeAdditionalInfo`	Contains additional information regarding the charged party.

8.4 TpCallPartyToChargeAdditionalInfo

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	NULL	`Undefined`
P_CALL_PARTY_SPECIAL	TpAddress	`CallPartySpecial`

8.5 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

~~8.6TpCallChargeOrder~~

~~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_TRANSPARENT~~	~~TpOctetSet~~	~~TransparentCharge~~
~~P_CALL_CHARGE_PREDEFINED_SET~~	~~TpInt32~~	~~ChargePlan~~

~~8.7~~8.6 TpCallChargeOrderCategory

Defines the type of charging to be applied

Name	Value	Description
P_CALL_CHARGE_TRANSPARENT	0	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	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).

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04 CR 061 ⌘rev - ⌘ Current version: 4.5.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title:	⌘	Correction to User Interaction Prepaid Sequence Diagrams
Source:	⌘	N5
Work item code:	⌘	OSA1
Date:	⌘	31/10/2002
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)
Rel-6 (Release 6)

Reason for change:	⌘	The description of the Prepaid and Prepaid with Advice of Charge sequence diagrams in Generic Call Control is incorrect. They both indicate that an announcement is played only to party A in a call controlled by a GCC application, when both A and B parties are connected. The announcement will in fact be played to both parties, since there is no means in GCC to separate the two parties in the call. This error has been partially corrected in GCC for Release 5 (N5-020500). This CR introduces the changes made in N5-020500 for Release 4, and completes them.
Summary of change:	⌘	Change the Prepaid and Prepaid with Advice of Charge sequence diagrams to indicate that the announcement is played to both parties.
Consequences if not approved:	⌘	Developers use these sequence diagrams as examples of how OSA/Parlay really behaves. Since they consider that these examples are provided by the real experts, they consider they must be right and should be followed. If we don't correct such errors, we are deliberately misleading developers, and can only expect interoperability problems at later stages.

Clauses affected:	⌘	6.1.11, 6.1.12

Other specs affected: ⌘	Y	N		⌘
		X	Other core specifications	
		X	Test specifications	
		X	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/specs/CR.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.

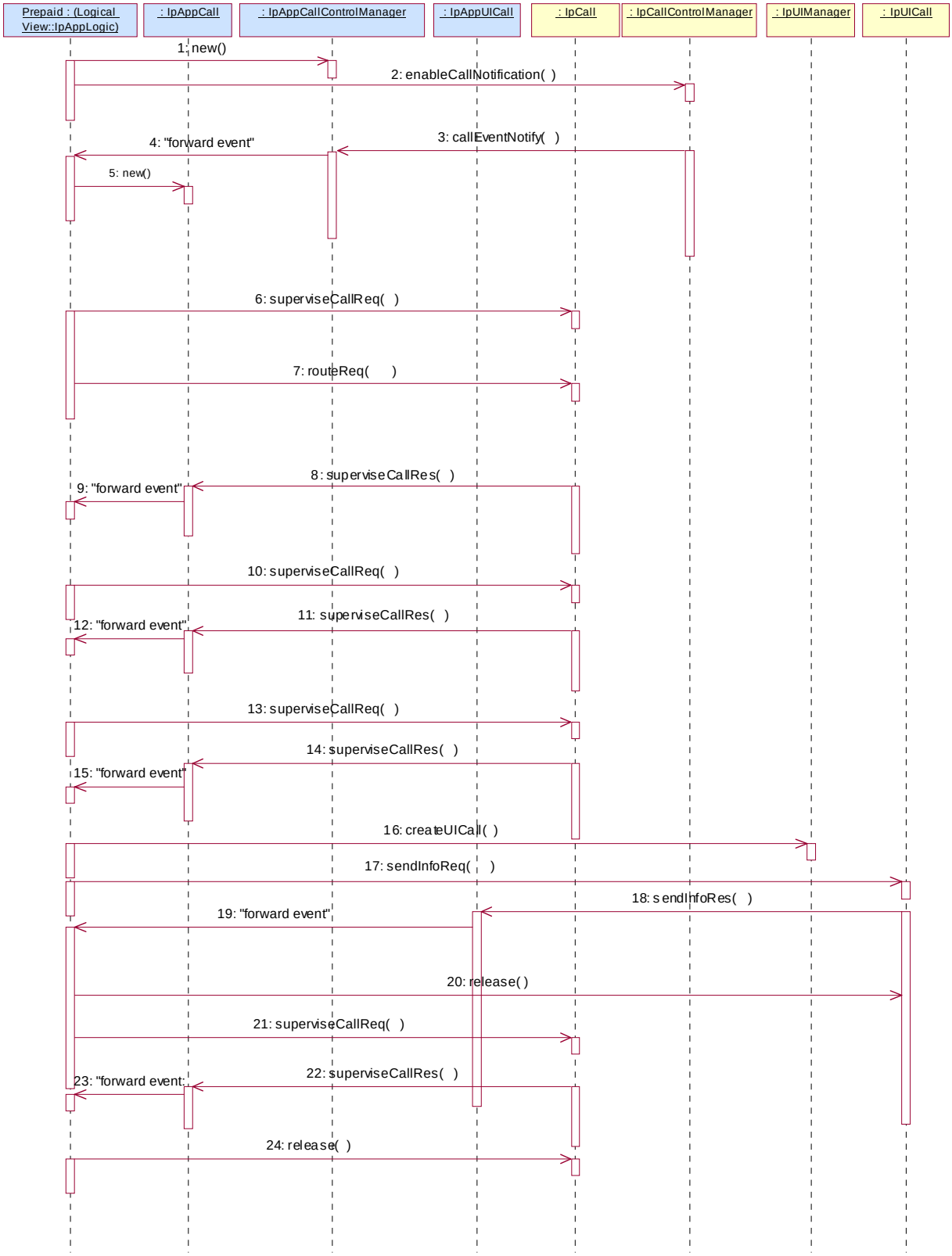
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest 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.

6.1.11 Prepaid

This sequence shows a Pre-paid application.

The subscriber is using a pre-paid card or credit card to pay for the call. The application each time allows a certain timeslice for the call. After the timeslice, a new timeslice can be started or the application can terminate the call. In the following sequence the end-user will receive an announcement before his final timeslice.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call,

that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: The incoming call triggers the Pre-Paid Application (PPA).

4: The message is forwarded to the application.

5: A new object on the application side for the Generic Call object is created

6: The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.

7: Before continuation of the call, PPA sends all charging information, a possible tariff switch time and the call duration supervision period, towards the GW which forwards it to the network.

8: At the end of each supervision period the application is informed and a new period is started.

9: The message is forwarded to the application.

10: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

11: At the end of each supervision period the application is informed and a new period is started.

12: The message is forwarded to the application.

13: The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.

14: When the user is almost out of credit ~~an announcement is played to inform about this. The announcement is played only to the leg of the A-party, the B-party will not hear the announcement~~the application is informed.

15: The message is forwarded to the application.

16: The application decides to play an announcement to the parties in this call. A new UICall object is created and associated with the call~~controlling leg~~.

17: An announcement is played ~~to the controlling leg~~ informing the user about the near-expiration of his credit limit. ~~The B-subscriber will not hear the announcement.~~

18: When the announcement is completed the application is informed.

19: The message is forwarded to the application.

20: The application releases the UICall object.

21: The user does not terminate so the application terminates the call after the next supervision period.

22: The supervision period ends

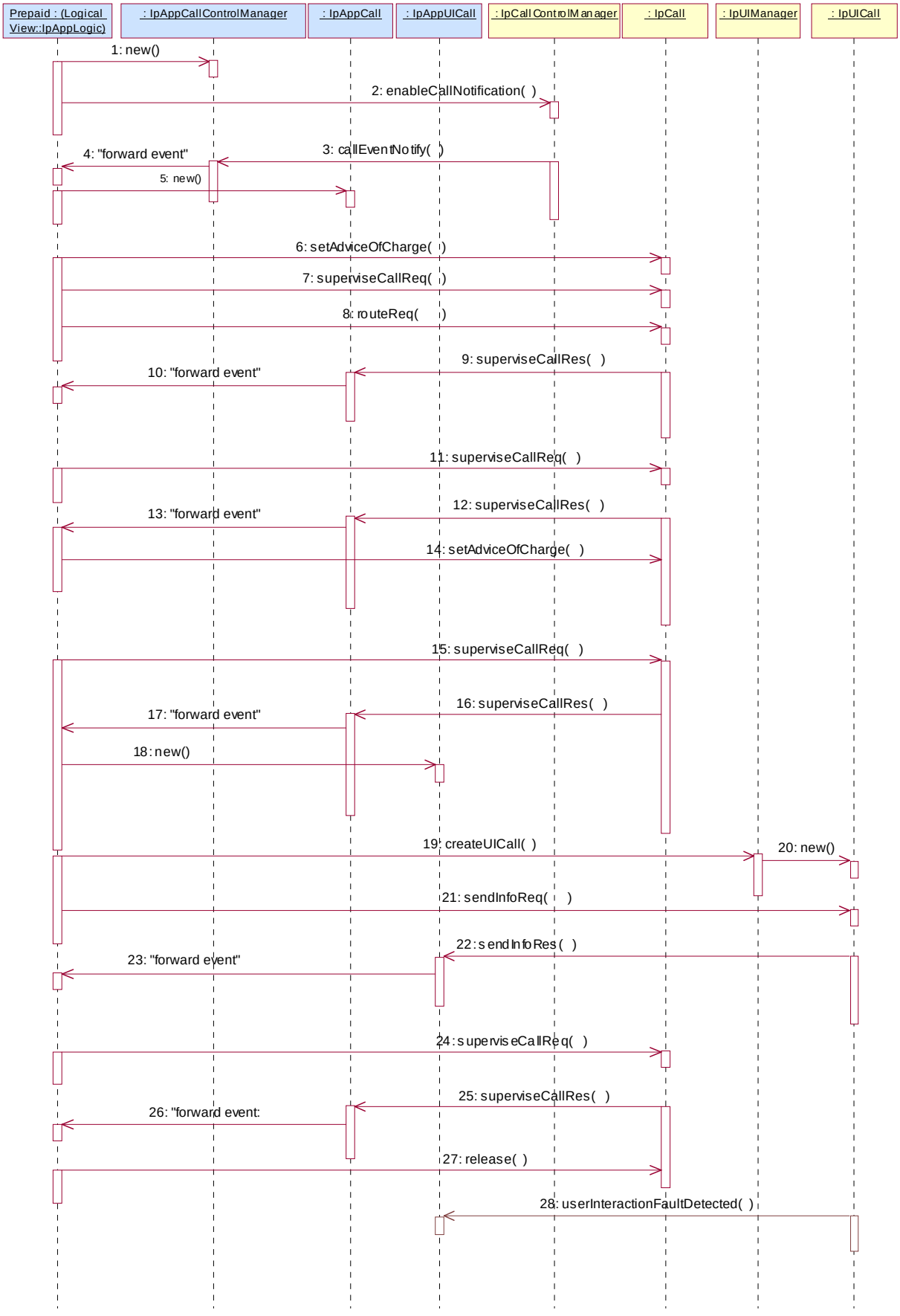
23: The event is forwarded to the logic.

24: The application terminates the call. Since the user interaction is already explicitly terminated no userInteractionFaultDetected is sent to the application.

6.1.12 Pre-Paid with Advice of Charge (AoC)

This sequence shows a Pre-paid application that uses the Advice of Charge feature.

The application will send the charging information before the actual call setup and when during the call the charging changes new information is sent in order to update the end-user. Note: the Advice of Charge feature requires an application in the end-user terminal to display the charges for the call, depending on the information received from the application.


Prepaid : (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpAppUICall
: IpCallControlManager
: IpCall
: IpUIManager
: IpUICall
1: new()
2: enableCallNotification( )
3: callEventNotify( )
4: "forward event"
5: new()
6: setAdviceOfCharge( )
7: superviseCallReq( )
8: routeReq( )
9: superviseCallRes( )
10: "forward event"
11: superviseCallReq( )
12: superviseCallRes( )
13: "forward event"
14: setAdviceOfCharge( )
15: superviseCallReq( )
16: superviseCallRes( )
17: "forward event"
18: new()
19: createUICall( )
20: new()
21: sendInfoReq( )
22: sendInfoRes( )
23: "forward event"
24: superviseCallReq( )
25: superviseCallRes( )
26: "forward event:
27: release( )
28: userInteractionFaultDetected( )

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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: The incoming call triggers the Pre-Paid Application (PPA).

4: The message is forwarded to the application.

5: A new object on the application side for the Call object is created

6: The Pre-Paid Application (PPA) sends the AoC information (e.g. the tariff switch time). (it shall be noted the PPA contains ALL the tariff information and knows how to charge the user).

During this call sequence 2 tariff changes take place. The call starts with tariff 1, and at the tariff switch time (e.g., 18:00 hours) switches to tariff 2. The application is not informed about this (but the end-user is!)

7: The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.

8: The application requests to route the call to the destination address.

9: At the end of each supervision period the application is informed and a new period is started.

10: The message is forwarded to the application.

11: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

12: At the end of each supervision period the application is informed and a new period is started.

13: The message is forwarded to the application.

14: Before the next tariff switch (e.g., 19:00 hours) the application sends a new AOC with the tariff switch time. Again, at the tariff switch time, the network will send AoC information to the end-user.

15: The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.

16: When the user is almost out of credit the application is informed~~an announcement is played to inform about this (19-21). The announcement is played only to the leg of the A-party, the B-party will not hear the announcement~~.

17: The message is forwarded to the application.

18: The application creates a new call back interface for the User interaction messages.

19: A new UI Call object that will handle playing of the announcement needs to be created

20: The Gateway creates a new UI call object that will handle playing of the announcement.

21: With this message the announcement is played to the parties in the call~~calling party~~.

22: The user indicates that the call should continue.

23: The message is forwarded to the application.

24: The user does not terminate so the application terminates the call after the next supervision period.

25: The user is out of credit and the application is informed.

26: The message is forwarded to the application.

27: With this message the application requests to release the call.

28: Terminating the call which has still a UICall object associated will result in a userInteractionFaultDetected. The UICall object is terminated in the gateway and no further communication is possible between the UICall and the application.

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04 CR 060 ⌘rev - ⌘ Current version: 4.5.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Correction to Sequence Diagrams to remove incorrect Framework references
Source:	⌘	N5
Work item code:	⌘	OSA1 ***Date:*** ⌘ 31/10/2002
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)
Rel-6 (Release 6)

Reason for change:	⌘	This CR corrects indications that call-control related events are received on the Framework, when in fact they are not.
Summary of change:	⌘	Change references to events being received by the Framework, where they are received by Call Control Manager interfaces.
Consequences if not approved:	⌘	Developers use these sequence diagrams as examples of how OSA/Parlay really behaves. Since they consider that these examples are provided by the real experts, they consider they must be right and should be followed. If we don't correct such errors, we are deliberately misleading developers, and can only expect interoperability problems at later stages.

Clauses affected:	⌘	6.1.4, 6.1.5, 6.1.6, 6.1.7, 6.1.8, 6.1.9, 7.1.3, 7.1.5

Other specs affected: ⌘	Y	N		⌘
		X	Other core specifications	
		X	Test specifications	
		X	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/specs/CR.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.

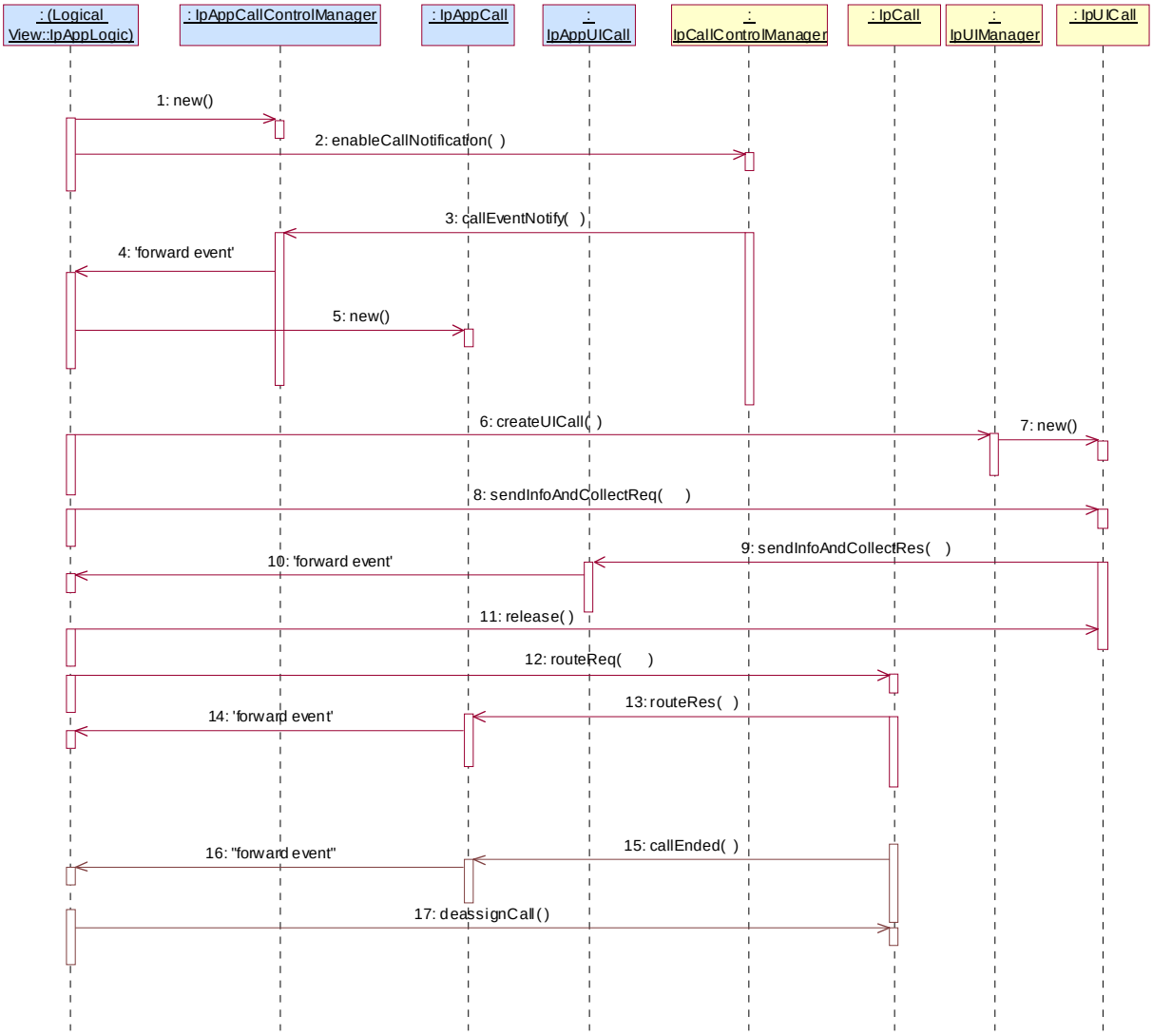
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest 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.

←========== FIRST MODIFIED SECTION ==========→

6.1.4 Call Barring 1

The following sequence diagram shows a call barring service, initiated as a result of a prearranged event being received by the call control service~~framework~~. Before the call is routed to the destination number, the calling party is asked for a PIN code. The code is accepted and the call is routed to the original called party.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range prompted for a password before the call is allowed to progress. When a new call, that matches the event criteria set, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

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

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

6: This message is used to create a new UICall object. The reference to the call object is given when creating the UICall.

7: Provided all the criteria are fulfilled, a new UICall object is created.

8: The call barring service dialogue is invoked.

9: The result of the dialogue, which in this case is the PIN code, is returned to its callback object.

10: This message is used to forward the previous message to the IpAppLogic.

11: This message releases the UICall object.

12: Assuming the correct PIN is entered, the call is forward routed to the destination party.

13: This message passes the result of the call being answered to its callback object.

14: This message is used to forward the previous message to the IpAppLogic

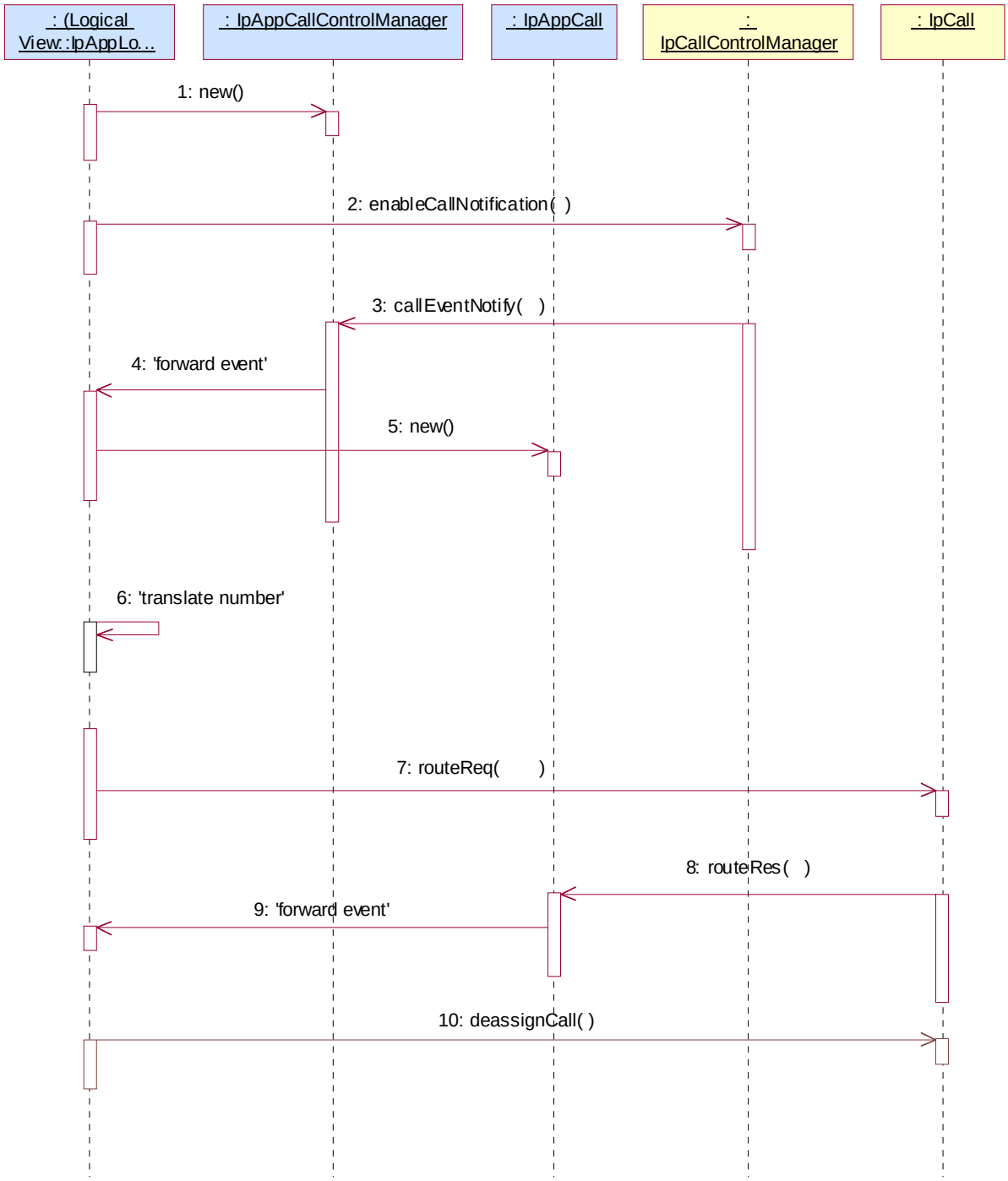
15: When the call is terminated in the network, the application will receive a notification. This notification will always be received when the call is terminated by the network in a normal way, the application does not have to request this event explicitly.

16: The event is forwarded to the application.

17: The application must free the call related resources in the gateway by calling deassignCall.

6.1.5 Number Translation 1

The following sequence diagram shows a simple number translation service, initiated as a result of a prearranged event being received by the call control service~~framework~~.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria set in message 2, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

4: This message is used to forward message 3 to the IpAppLogic.

5: This message is used by the application to create an object implementing the IpAppCall interface. The reference to this object is passed back to the object implementing the IpCallControlManager using the return parameter of message 3.

6: This message invokes the number translation function.

7: The returned translated number is used in message 7 to route the call towards the destination.

8: This message passes the result of the call being answered to its callback object

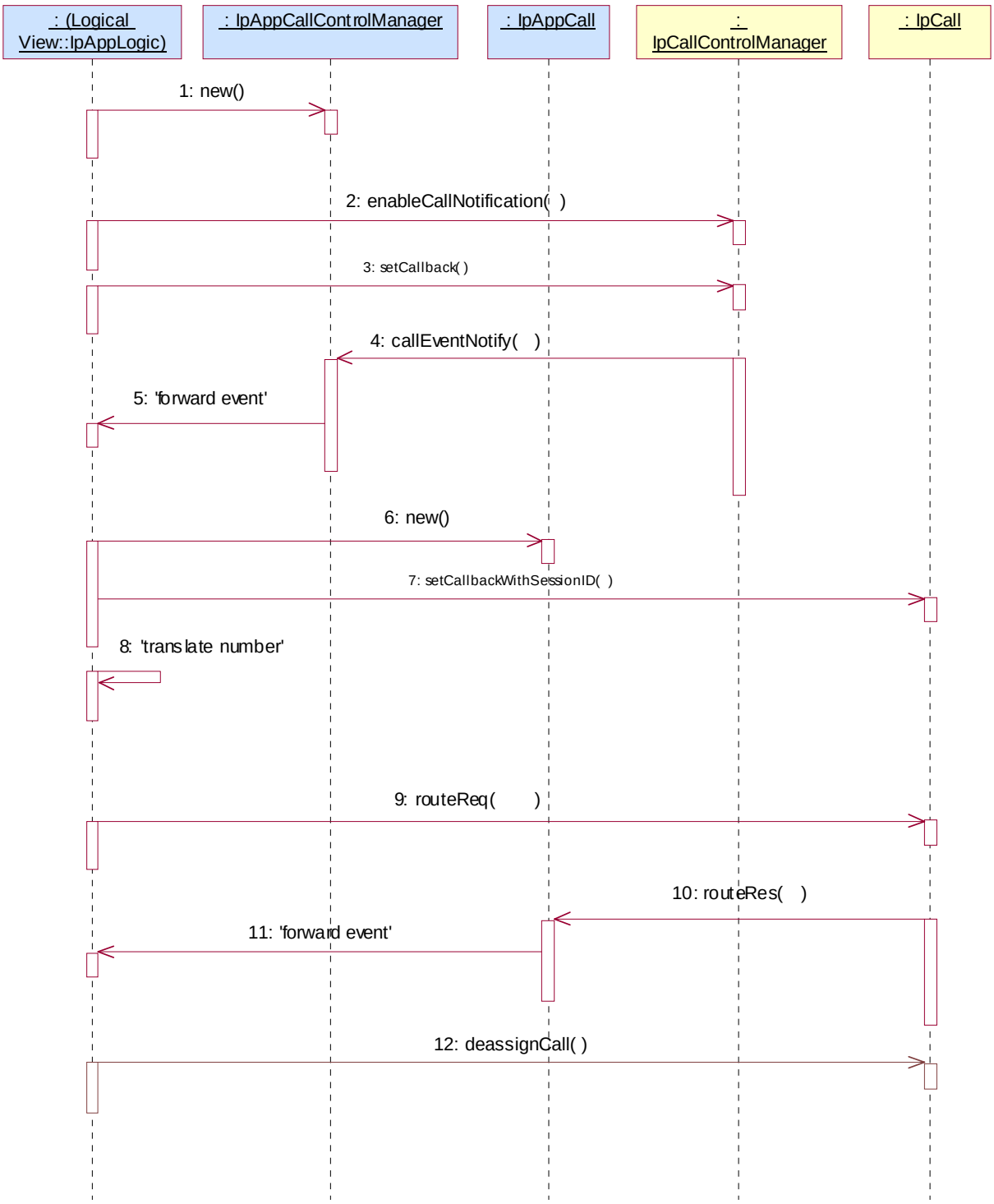
9: This message is used to forward the previous message to the IpAppLogic.

10: The application is no longer interested in controlling the call and therefore deassigns the call. The call will continue in the network, but there will be no further communication between the call object and the application.

6.1.6 Number Translation 1 (with callbacks)

The following sequence diagram shows a simple number translation service, initiated as a result of a prearranged event being received by the call control service~~framework~~.

For illustration, in this sequence the callback references are set explicitly. This is optional. All the callbacks references can also be passed in other methods. From an efficiency point of view that is also the preferred method. The rest of the sequences use that mechanism.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria set in message 2, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message sets the reference of the IpAppCallControlManager object in the CallControlManager. The CallControlManager reports the callEventNotify to referenced object only for enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification.

4: This message is used to pass the new call event to the object implementing the IpAppCallControlManager 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 IpAppCall interface.

7: This message is used to set the reference to the IpAppCall for this call.

8: This message invokes the number translation function.

9: The returned translated number is used in message 7 to route the call towards the destination.

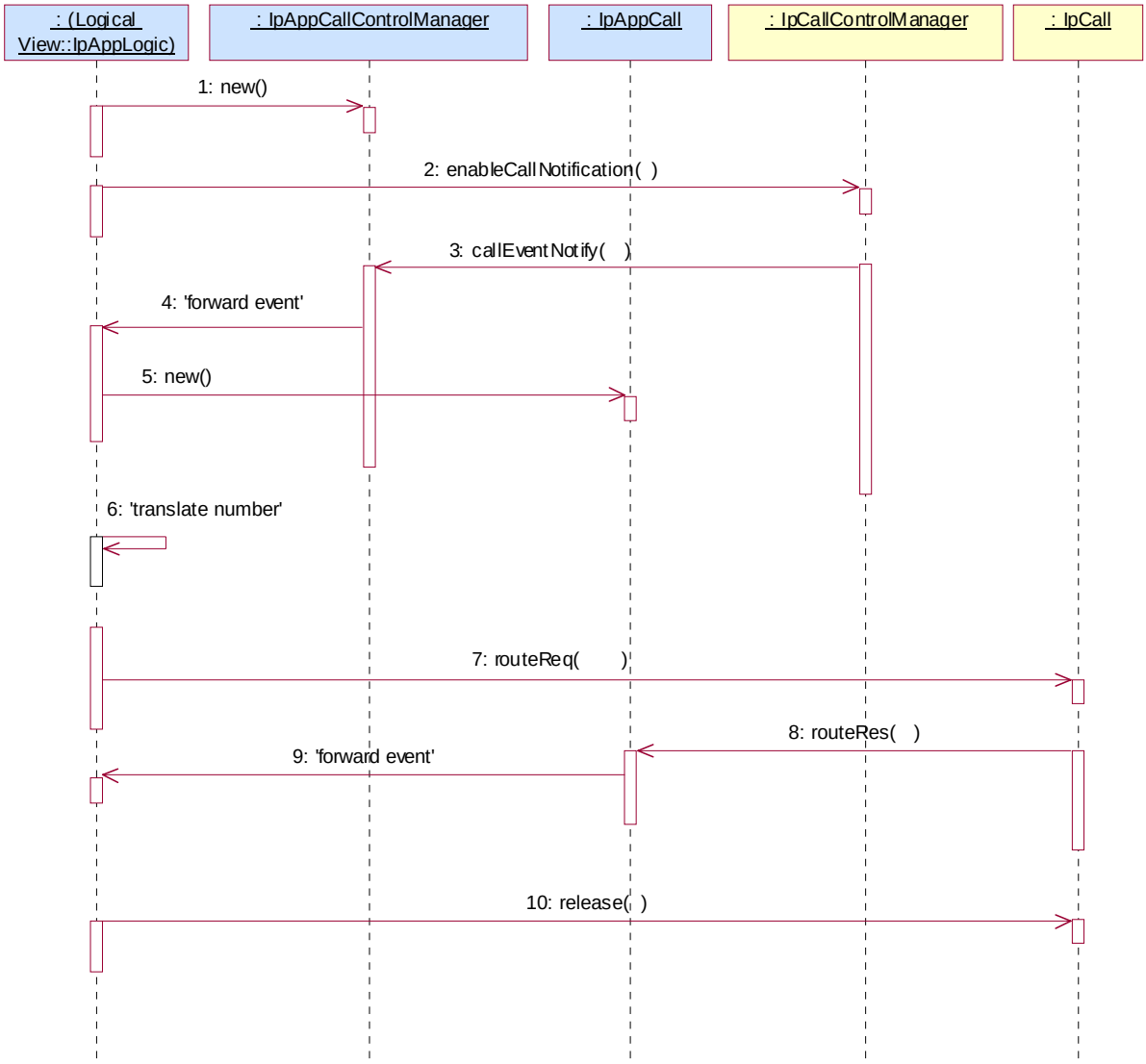
10: This message passes the result of the call being answered to its callback object

11: This message is used to forward the previous message to the IpAppLogic.

12: The application is no longer interested in controlling the call and therefore deassigns the call. The call will continue in the network, but there will be no further communication between the call object and the application.

6.1.7 Number Translation 2

The following sequence diagram shows a number translation service, initiated as a result of a prearranged event being received by the call control service~~framework~~. If the translated number being routed to does not answer or is busy then the call is automatically released.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

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

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

6: This message invokes the number translation function.

7: The returned translated number is used to route the call towards the destination.

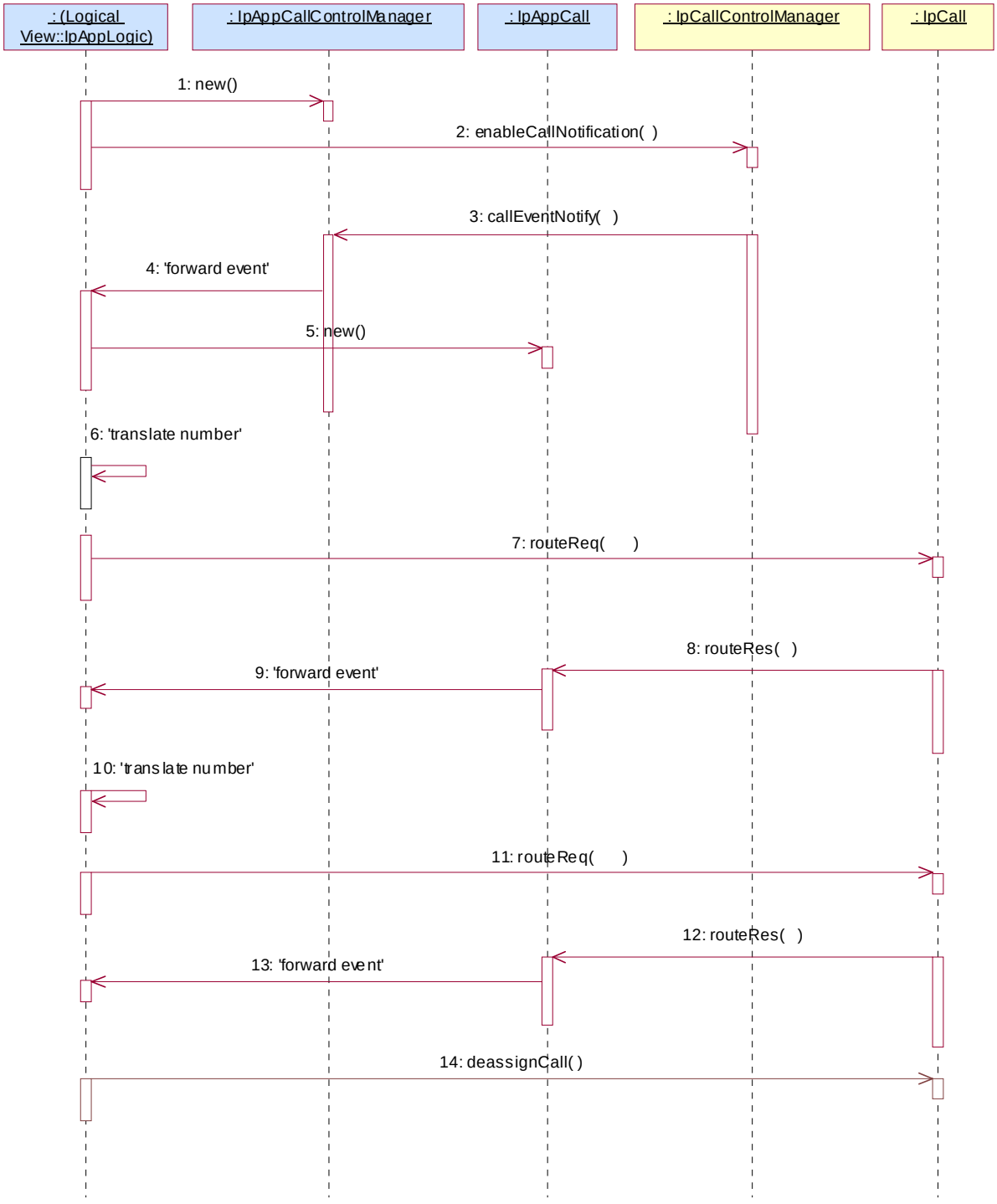
8: Assuming the called party is busy or does not answer, the object implementing the IpCall interface sends a callback in this message, indicating the unavailability of the called party.

9: This message is used to forward the previous message to the IpAppLogic.

10: The application takes the decision to release the call.

6.1.8 Number Translation 3

The following sequence diagram shows a number translation service, initiated as a result of a prearranged event being received by the call control service~~framework~~. If the translated number being routed to does not answer or is busy then the call is automatically routed to a voice mailbox.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

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

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

6: This message invokes the number translation function.

7: The returned translated number is used to route the call towards the destination.

8: Assuming the called party is busy or does not answer, the object implementing the IpCall interface sends a callback, indicating the unavailability of the called party.

9: This message is used to forward the previous message to the IpAppLogic.

10: The application takes the decision to translate the number, but this time the number is translated to a number belonging to a voice mailbox system.

11: This message routes the call towards the voice mailbox.

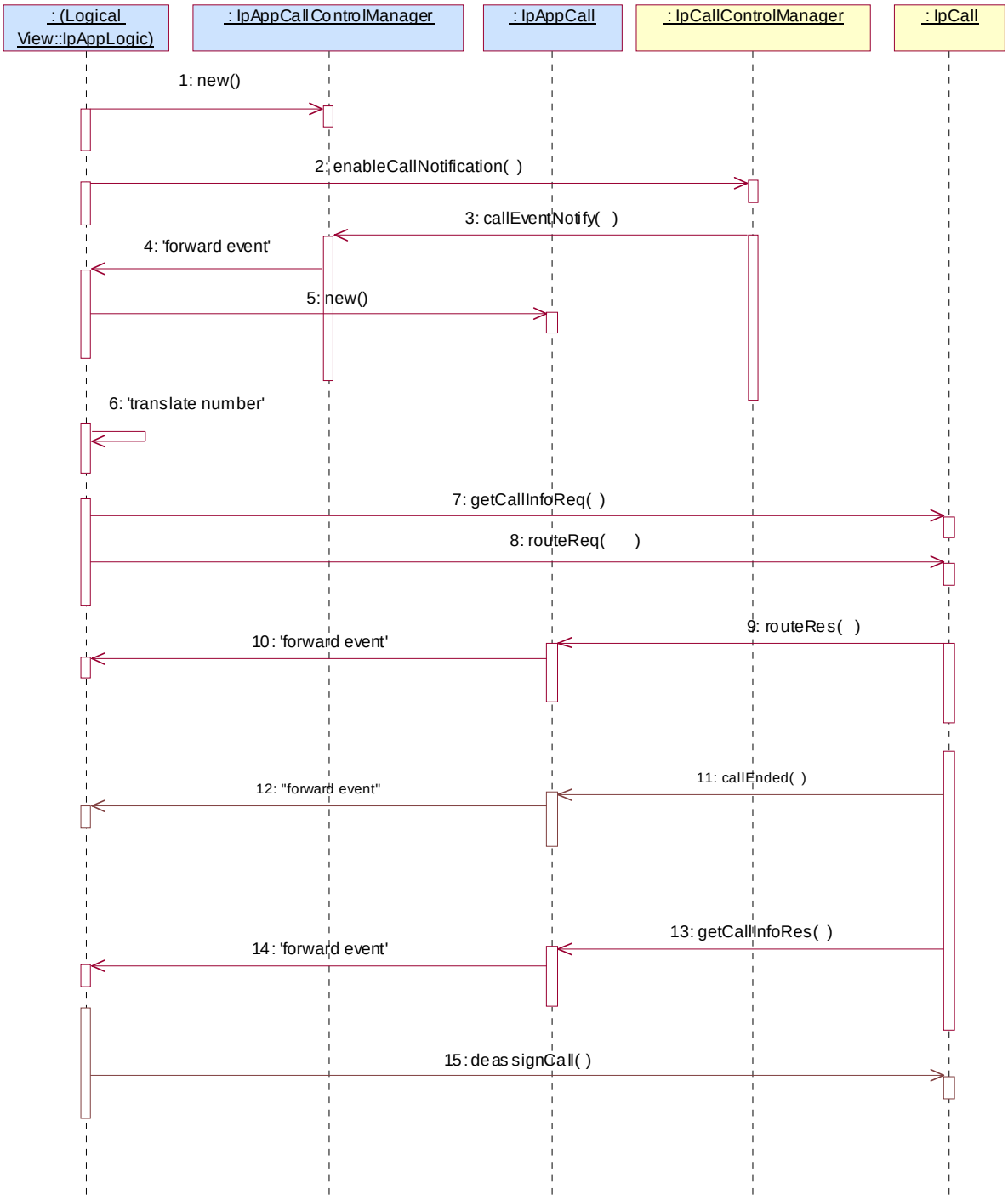
12: This message passes the result of the call being answered to its callback object.

13: This message is used to forward the previous message to the IpAppLogic.

14: The application is no longer interested in controlling the call and therefore deassigns the call. The call will continue in the network, but there will be no further communication between the call object and the application.

6.1.9 Number Translation 4

The following sequence diagram shows a number translation service, initiated as a result of a prearranged event being received by the call control service~~framework~~. Before the call is routed to the translated number, the application requests for all call related information to be delivered back to the application on completion of the call.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

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

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

6: This message invokes the number translation function.

7: The application instructs the object implementing the IpCall interface to return all call related information once the call has been released.

8: The returned translated number is used to route the call towards the destination.

9: This message passes the result of the call being answered to its callback object.

10: This message is used to forward the previous message to the IpAppLogic.

11: Towards the end of the call, when one of the parties disconnects, a message (not shown) is directed to the object implementing the IpCall. This causes an event, to be passed to the object implementing the IpAppCall object.

12: This message is used to forward the previous message to the IpAppLogic.

13: The application now waits for the call information to be sent. Now that the call has completed, the object implementing the IpCall interface passes the call information to its callback object.

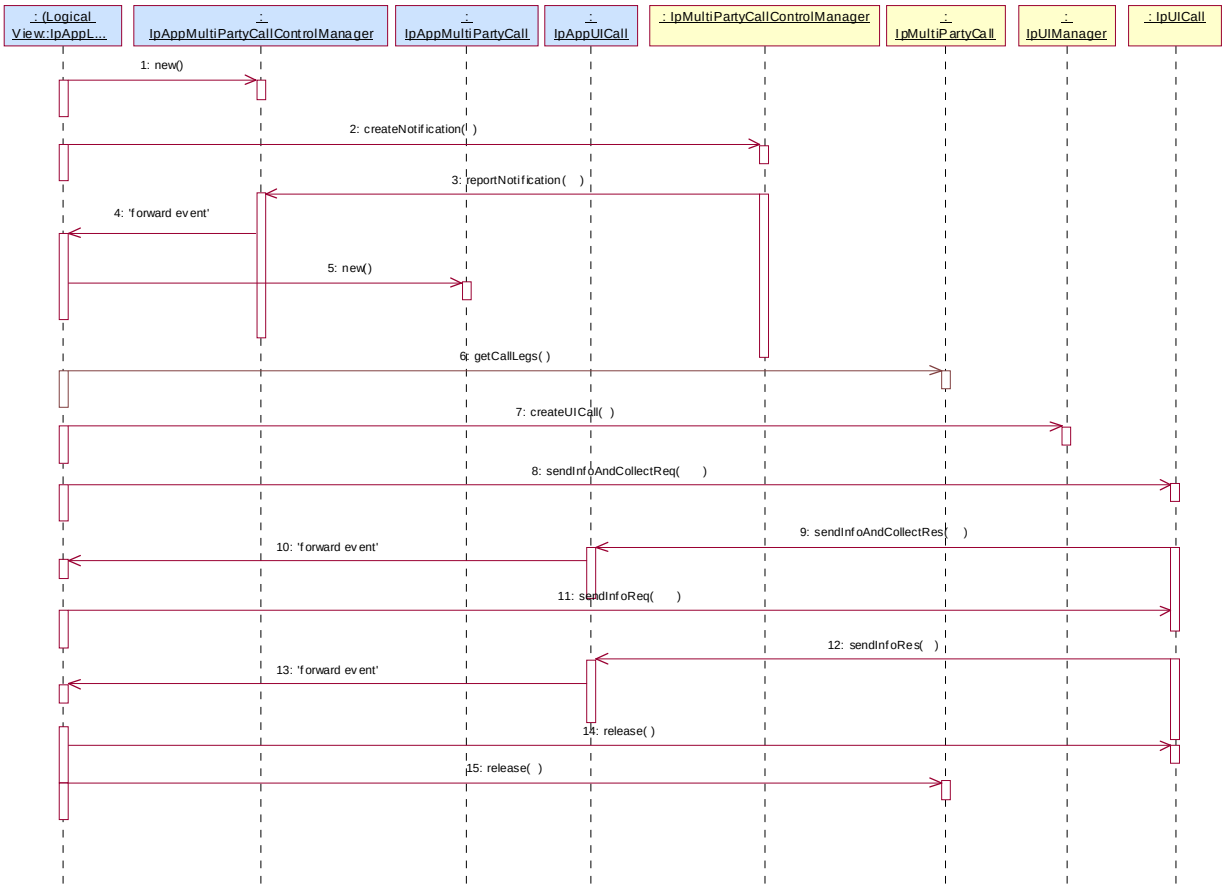
14: This message is used to forward the previous message to the IpAppLogic

15: After the last information is received, the application deassigns the call. This will free the resources related to this call in the gateway.

←========== NEXT MODIFIED SECTION ==========→

7.1.3 Call Barring 2

The following sequence diagram shows a call barring service, initiated as a result of a prearranged event being received by the call control service~~framework~~. Before the call is routed to the destination number, the calling party is asked for a PIN code. The code is rejected and the call is cleared.



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. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range prompted for a password before the call is allowed to progress. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface.

4: This message is used to forward message 3 to the IpAppLogic.

5: 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.

6: The application requests an list of all the legs currently in the call.

7: This message is used to create a UICall object that is associated with the incoming leg of the call.

8: The call barring service dialogue is invoked.

9: The result of the dialogue, which in this case is the PIN code, is returned to its callback object.

10: This message is used to forward the previous message to the IpAppLogic

11: Assuming an incorrect PIN is entered, the calling party is informed using additional dialogue of the reason why the call cannot be completed.

12: This message passes the indication that the additional dialogue has been sent.

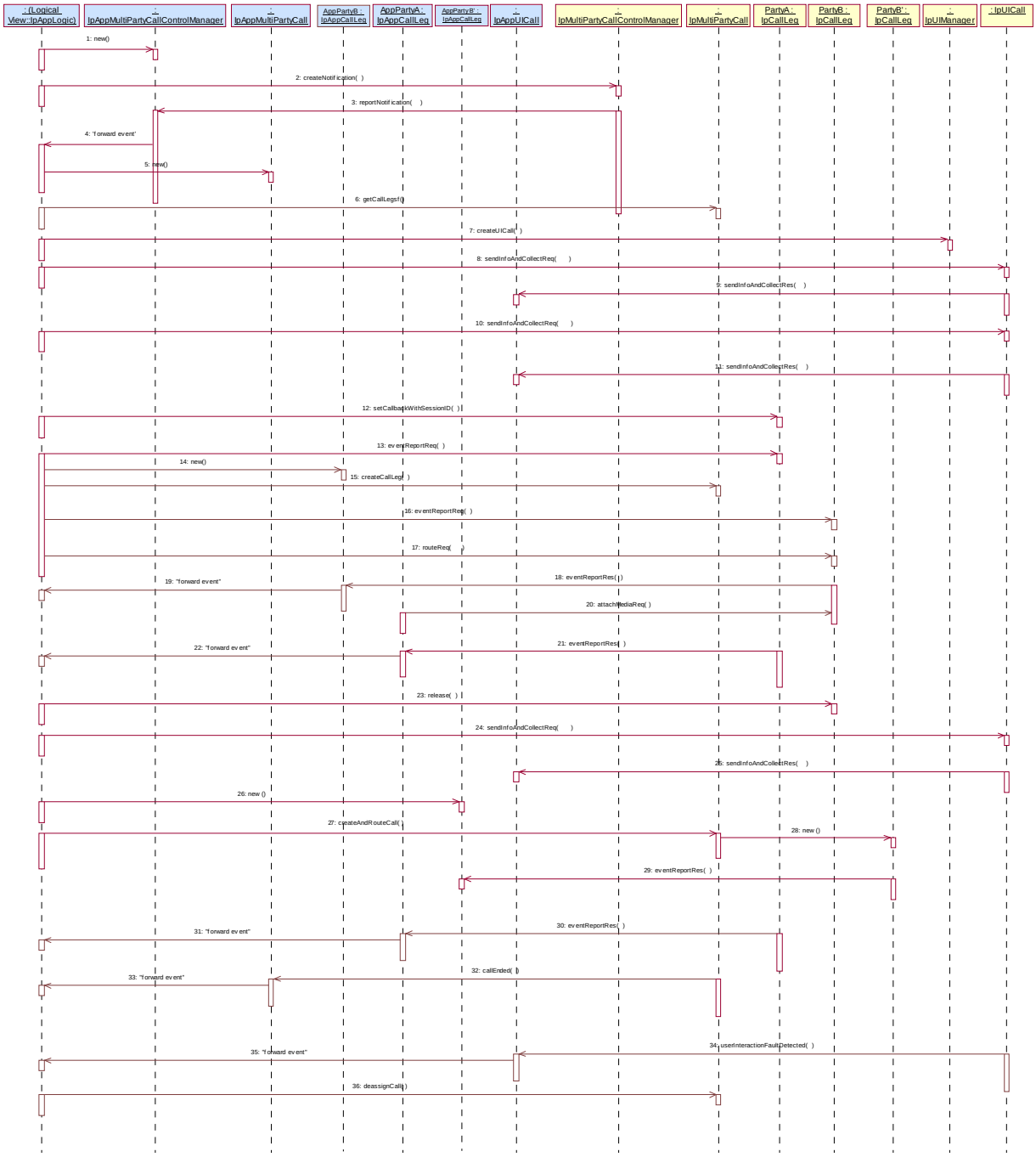
13: This message is used to forward the previous message to the IpAppLogic.

14: No more UI is required, so the UICall object is released.

15: This message is used by the application to clear the call.

7.1.5 Complex Card Service

The following sequence diagram shows an advanced card service, initiated as a result of a prearranged event being received by the call control service~~framework~~. Before the call is made, the calling party is asked for an ID and PIN code. If the ID and PIN code are accepted, the calling party is prompted to enter the address of the destination party. A trigger of '#5' is then set on the controlling leg (the calling party's leg) such that if the calling party enters a '#5' an event will be sent to the application. The call is then routed to the destination party. 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.



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. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range result in the caller being prompted for a password before the call is allowed to progress. When a new call, that matches the event criteria set in message 2, arrives a message (not shown) is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface.

4: This message is used to forward message 3 to the IpAppLogic.

5: 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 message 3.

6: This message returns the call legs currently in the call. In principle a reference to the call leg of the calling party is already obtained by the application when it was notified of the new call event.

7: This message is used to associate a user interaction object with the calling party.

8: The initial card service dialogue is invoked using this message.

9: The result of the dialogue, which in this case is the ID and PIN code, is returned to its callback object using this message and eventually forwarded via another message (not shown) to the IpAppLogic.

10: Assuming the correct ID and PIN are entered, the final dialogue is invoked.

11: The result of the dialogue, which in this case is the destination address, is returned and eventually forwarded via another message (not shown) to the IpAppLogic.

12: This message is used to forward the address of the callback object.

13: The trigger for follow-on calls is set (on service code).

14: A new AppCallLeg is created to receive callbacks for another leg. Alternatively, the already existing AppCallLeg object could be passed in the subsequent createCallLeg(). In that case the application has to use the sessionIDs of the legs to distinguish between callbacks destined for the A-leg and callbacks destined for the B-leg.

15: 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.

16: The application requests to be notified when the leg is answered.

17: The application routes the leg. As a result the network will try to reach the associated party.

18: When the B-party answers the call, the application is notified.

19: The event is forwarded to the application logic.

20: Legs that are created and routed explicitly are by default in state detached. This means that the media is not connected to the other parties in the call. In order to allow inband communication between the new party and the other parties in the call the media have to be explicitly attached.

21: At some time during the call the calling party enters '#5'. This causes this message to be sent to the object implementing the IpAppCallLeg interface, which forwards this event as a message (not shown) to the IpAppLogic.

22: The event is forwarded to the application.

23: This message releases the called party.

24: Another user interaction dialogue is invoked.

25: The result of the dialogue, which in this case is the new destination address is returned and eventually forwarded via another message (not shown) to the IpAppLogic.

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

27: The call is then forward routed to the new destination party.

28: As a result a new Callleg object is created.

29: This message passes the result of the call being answered to its callback object and is eventually forwarded via another message (not shown) to the IpAppLogic.

30: When the A-party terminates the application is informed.

31: The event is forwarded to the application logic.

32: Since the release of the A-party will in this case terminate the entire call, the application is also notified with this message.

33: The event is forwarded to the application logic.

34: Since the user interaction object were not released at the moment that the call terminated, the application receives this message to indicate that the UI resources are released in the gateway and no further communication is possible.

35: The event is forwarded to the application logic.

36: The application deassigns the call object.

←========== END MODIFIED SECTION ==========→

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04** **CR** **059** ⌘ **rev** **-** ⌘ Current version: **4.5.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction to TpReleaseCauseSet in Multi Party Call Control
Source:	⌘	N5
Work item code:	⌘	OSA1
Date:	⌘	31/10/2002
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)
Rel-6 (Release 6)

Reason for change:	⌘	The definition of TpReleaseCauseSet in TS 29.198-04 contains an error: It is defined as being a Numbered Set of Data Elements of TpCallReleaseCause. But TpCallReleaseCause is the GCC release cause data type, and TpReleaseCauseSet is used for MPCC, so should be a Numbered Set of Data Elements of TpReleaseCause. The IDL has the correct definition.
Summary of change:	⌘	Change datatype to match correct IDL description of TpReleaseCauseSet
Consequences if not approved:	⌘	A contradiction will exist between the IDL and the Word document, and the Word document will mislead developers. If no alignment is made, some developers will chose one implementation of the type, others the other, and interworking problems will arise.

Clauses affected:	⌘	Annex A			
Other specs affected:	⌘	**Y**	**N**		⌘
			X	Other core specifications	
			X	Test specifications	
			X	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/specs/CR.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.

←========== MODIFIED SECTION ==========→

7.6.2.34 TpReleaseCause

Defines the reason for which a call is released.

Name	Value	Description
P_UNDEFINED	0	The reason of release is not known, because no info was received from the network.
P_USER_NOT_AVAILABLE	1	The user is not available in the network. This means that the number is not allocated or that the user is not registered.
P_BUSY	2	The user is busy.
P_NO_ANSWER	3	No answer was received
P_NOT_REACHABLE	4	The user terminal is not 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	A disconnect was received.
P_CALL_RESTRICTED	8	The call was subject of restrictions
P_UNAVAILABLE_RESOURCE	9	The request could not be carried out as no resources were available.
P_GENERAL_FAILURE	10	A general network failure occurred.
P_TIMER_EXPIRY	11	The call / call leg was released because an activity timer expired.

7.6.2.35 TpReleaseCauseSet

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

←========== END MODIFIED SECTION ==========→

Annex A (normative): OMG IDL Description of Call Control SCF

The OMG IDL representation of this interface specification is contained in text files (contained in archive 2919804IDL.ZIP) which accompany the present document.

In file mpcc_data.idl:

```
typedef sequence <TpReleaseCause> TpReleaseCauseSet;
```

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04 CR 058 ⌘rev - ⌘ Current version: 4.5.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title: ⌘	Correction of status of methods to interfaces in clause 6.3	
Source: ⌘	N5	
Work item code: ⌘	OSA1	**Date:** ⌘ 27/09/2002
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)*
Rel-6	*(Release 6)*

Reason for change: ⌘	There is no requirement in the standard about the necessity to implement all or only some of the methods defined for an interface.
Summary of change: ⌘	Clarify which methods are mandatory and which are optional.
Consequences if not approved: ⌘	Application developers will not know which methods will actually be available.

Clauses affected: ⌘	6.3 Generic Call Control Service Interface Classes

Other specs affected: ⌘

Y	N	
	X	Other core specifications
	X	Test specifications
	X	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/specs/CR.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 Call Control SCF

Two flavours of Call Control (CC) APIs have been included in 3GPP Release 4. These are the Generic Call Control (GCC) and the Multi-Party Call Control (MPCC). The GCC is the same API as was already present in the Release 99 specification (TS 29.198 v3.3.0) and is in principle able to satisfy the requirements on CC APIs for Release 4.

However, the joint work between 3GPP CN5, ETSI SPAN12 and the Parlay CC Working group with collaboration from JAIN has been focussed on the MPCC API. A number of improvements on CC functionality have been made and are reflected in this API. For this it was necessary to break the inheritance that previously existed between GCC and MPCC.

The joint CC group has furthermore decided that the MPCC is to be considered as the future base CC family and the technical work will not be continued on GCC. Errors or technical flaws will of course be corrected.

The following clauses describe each aspect of the CC Service Capability Feature (SCF).

The order is as follows:

- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause shows how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data definitions clause show a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part of this specification (29.198-2).

4.1 Call Model Description

The adopted call model has the following objects.

* 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.

4.2 General requirements on support of methods

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method.

Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

<========== NEXT MODIFIED SECTION ==========>

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.

For the generic call control service, only a subset of the call model defined in clause 4 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.

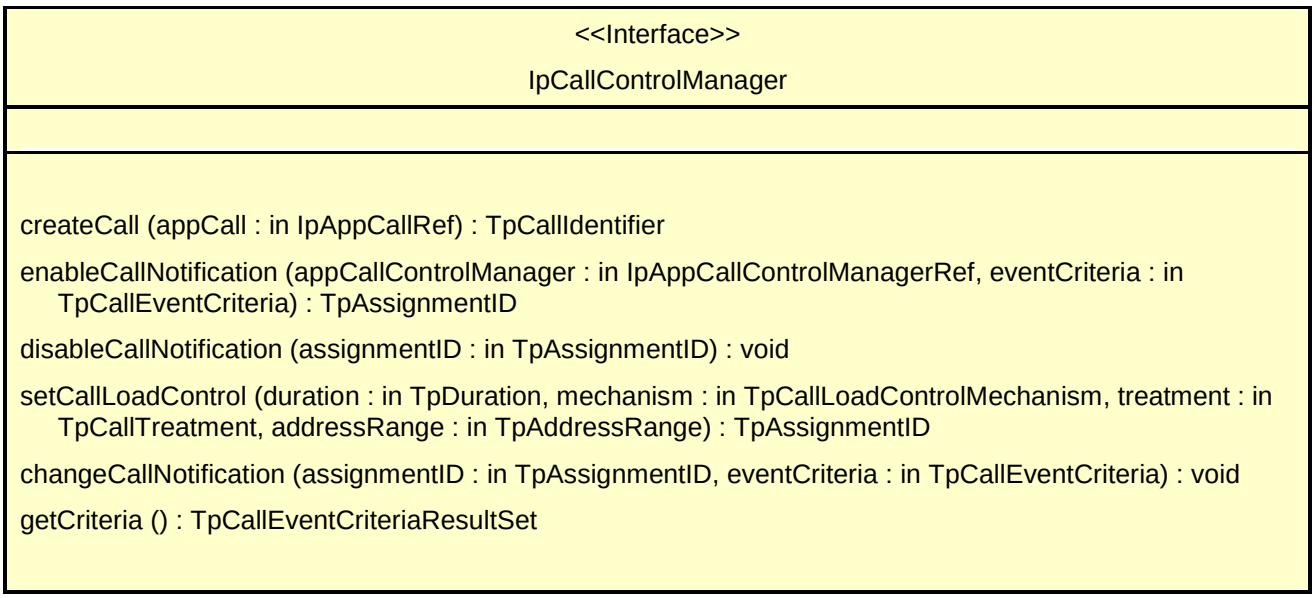
The GCCS is represented by the IpCallControlManager and IpCall interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppCallControlManager 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.

This interface shall be implemented by a Generic Call Control SCF. As a minimum requirement either the createCall() method shall be implemented, or the enableCallNotification() and disableCallNotification() methods shall be implemented.



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).

Returns callReference: Specifies the interface reference and sessionID of the call created.

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpCallIdentifier

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 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().

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

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.

Returns

TpAssignmentID

Raises

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 enableCallNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised. 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.

Returns assignmentID: Specifies the assignmentID assigned by the gateway to this request. This assignmentID can be used to correlate the callOverloadEncountered and callOverloadCeased methods with the request.

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.

Returns

TpAssignmentID

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 assignmentID 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.

Returns eventCriteria: Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Parameters

No Parameters were identified for this method

Returns

TpCallEventCriteriaResultSet

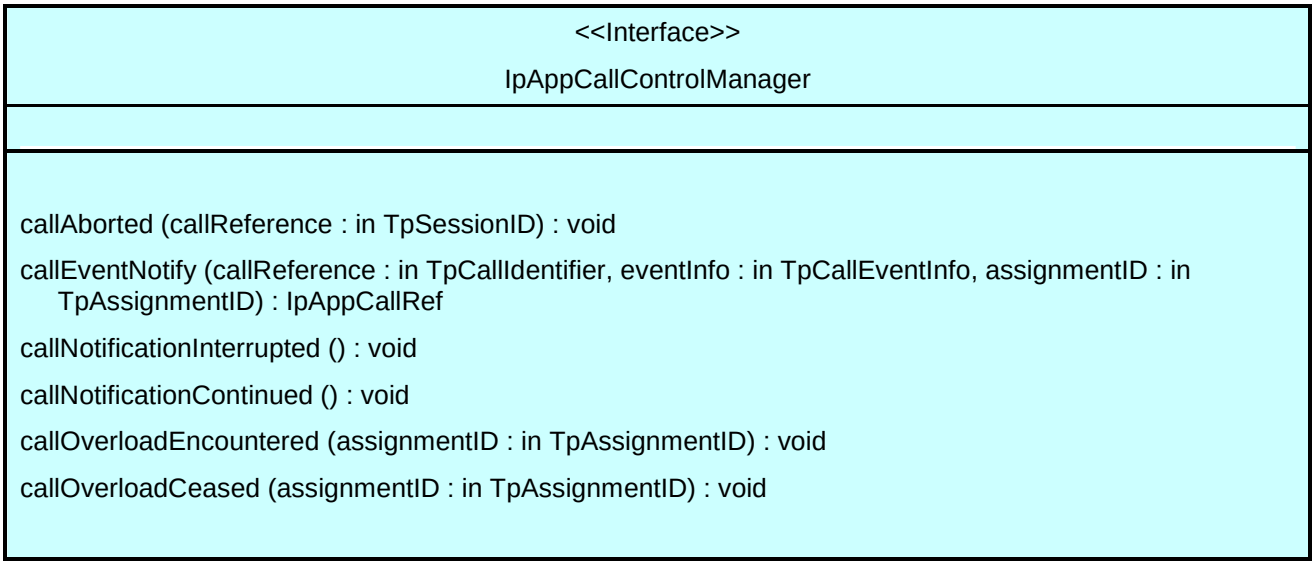
Raises

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.

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_CALL_MONITOR_MODE_INTERRUPT, 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_CALL_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.

Returns appCall: Specifies a reference to the application interface which implements the callback interface for the new call. If the application has previously explicitly passed a reference to the IpAppCall interface using a setCallback() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallback().

This parameter will be null if the notification is in NOTIFY mode.

Parameters

callReference : in TpCallIdentifier

Specifies the reference to the call interface to which the notification relates. If the notification is in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking callEventNotify may populate this parameter as it chooses.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

Specifies the assignment id which was returned by the enableCallNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

Returns

IpAppCallRef

Method

callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporarily 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

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 address range 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`

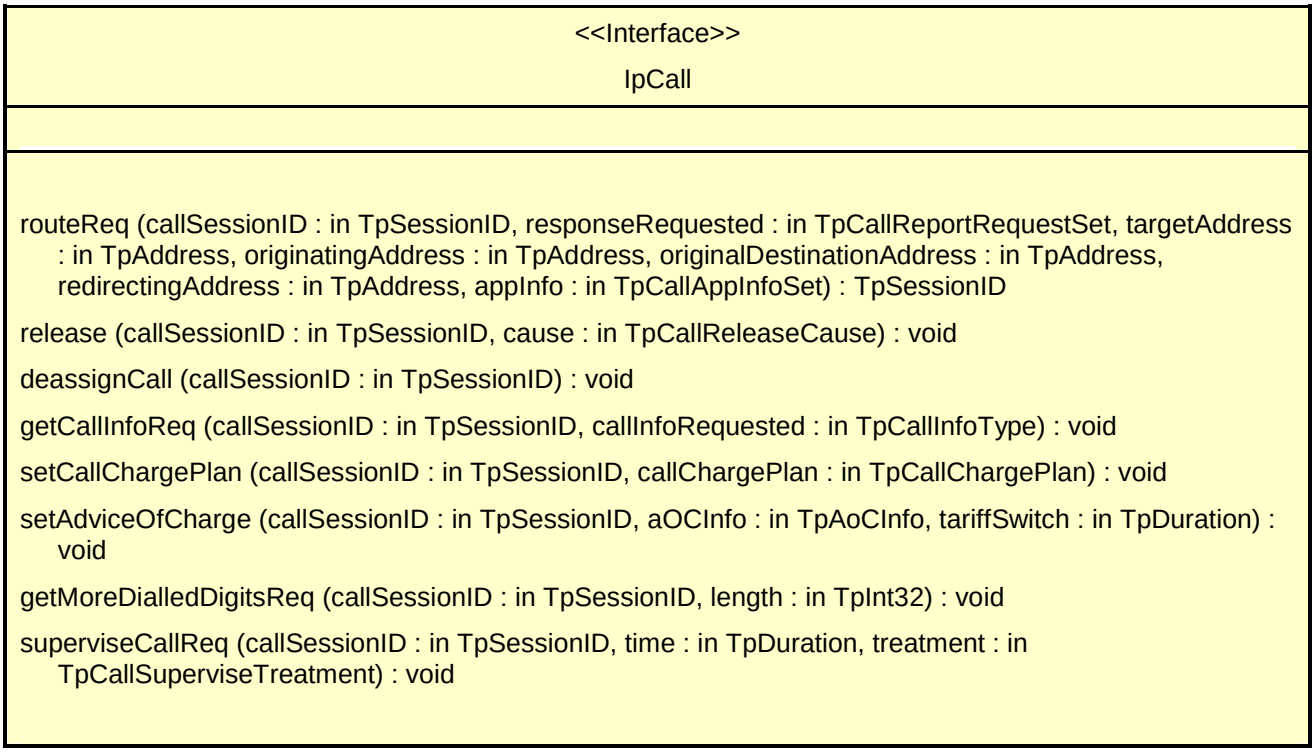
Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the address range for within which the overload has been ceased

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.

This interface shall be implemented by a Generic Call Control SCF. As a minimum requirement, the routeReq (), release() and deassignCall() methods shall be implemented.



Method
`routeReq()`

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

Note that in case of routeReq() it is recommended to request for 'successful' (e.g. 'answer' event) and 'failure' events at invocation, because those are needed for the application to keep track of the state of the call.

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.

Returns callLegSessionID: 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.

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).

Returns

TpSessionID

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.

A report 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

TpCommonExceptions, P_INVALID_SESSION_ID

Method

setCallChargePlan()

Set an operator specific charge plan for the call.

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

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.

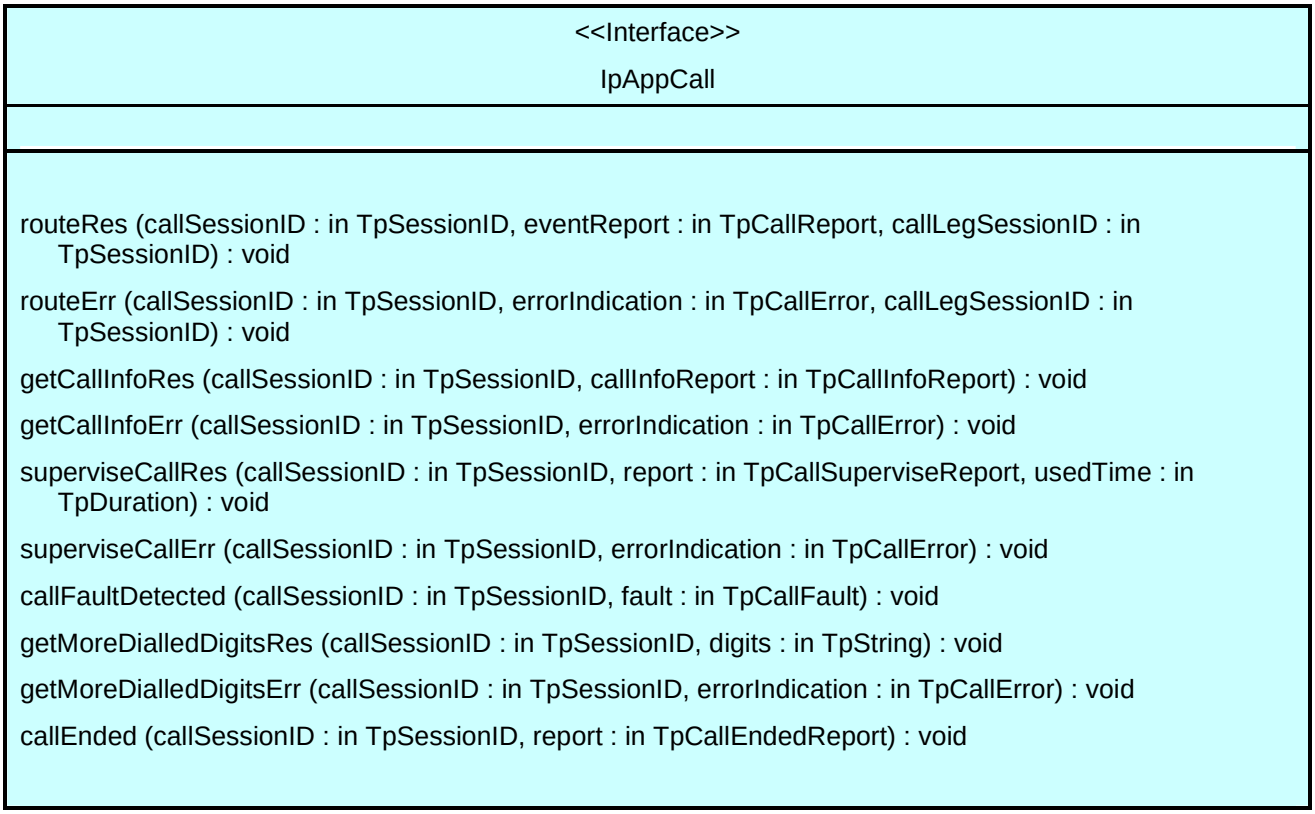
Raises

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.



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_CALL_MONITOR_MODE_INTERRUPT,

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 sessionID returned at the routeReq() and can be used to correlate the response with the request.

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.

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.

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.

Method
superviseCallRes()

This asynchronous method reports a call supervision event to the application when it has indicated its 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
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.

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.

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
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.

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3** **CR** **009** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction to TpReleaseCauseSet in Multi Party Call Control IDL
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	31/10/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)
Rel-6 (Release 6)

Reason for change:	⌘	The definition of TpReleaseCauseSet in TS 29.198-04-3 contains an error: It is defined as being a Numbered Set of Data Elements of TpCallReleaseCause. But TpCallReleaseCause is the GCC release cause data type, and TpReleaseCauseSet is used for MPCC, so should be a Numbered Set of Data Elements of TpReleaseCause. The IDL and WSDL have the correct definition.
Summary of change:	⌘	Change datatype to correct description of TpReleaseCauseSet
Consequences if not approved:	⌘	A contradiction will exist between the IDL, WSDL and the Word document, and the Word document will mislead developers. If no alignment is made, some developers will chose one implementation of the type, others the other, and interworking problems will arise.

Clauses affected:	⌘	Annex A, Annex B			
Other specs affected:	⌘	**Y**	**N**		⌘
			X	Other core specifications	
			X	Test specifications	
			X	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/specs/CR.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.

←========== MODIFIED SECTION ==========→

9.2.34 TpReleaseCause

Defines the reason for which a call is released.

Name	Value	Description
P_UNDEFINED	0	The reason of release is not known, because no info was received from the network.
P_USER_NOT_AVAILABLE	1	The user is not available in the network. This means that the number is not allocated or that the user is not registered.
P_BUSY	2	The user is busy.
P_NO_ANSWER	3	No answer was received
P_NOT_REACHABLE	4	The user terminal is not 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	A disconnect was received.
P_CALL_RESTRICTED	8	The call was subject of restrictions
P_UNAVAILABLE_RESOURCE	9	The request could not be carried out as no resources were available.
P_GENERAL_FAILURE	10	A general network failure occurred.
P_TIMER_EXPIRY	11	The call / call leg was released because an activity timer expired.
P_UNSUPPORTED_MEDIA	12	The call / call leg was released either because the message body of the request is in a format not supported or because the media is not supported.

9.2.35 TpReleaseCauseSet

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

←========== END MODIFIED SECTION ==========→

Annex A (normative): OMG IDL Description of Multi-Party Call Control SCF

The OMG IDL representation of this interface specification is contained in text files mpcc_data.idl and mpcc_interfaces.idl (contained in archive 291980403IDL.ZIP) which accompany the present document.

```
typedef sequence<TpReleaseCause> TpReleaseCauseSet;
```

Annex B (informative): W3C WSDL Description of Multi-Party Call Control SCF

The W3C WSDL representation of this specification is contained in text files (mpcc_data.wsdl, mpcc_interfaces.wsdl contained in archive 291980403WSDL.ZIP) which accompanies the present document.

```
<xsd:complexType name="TpReleaseCauseSet">
    <xsd:sequence>
        <xsd:element name="TpReleaseCauseSet" type="common_cc_dataxsd:TpReleaseCause" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04-2 CR 005 ⌘rev - ⌘ Current version: 5.1.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title:	⌘	Correction to TpCallEventCriteriaResult in Generic Call Control
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	31/10/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	The text description for TpCallEventCriteriaResult contains a name of an element in a struct data type which contradicts the IDL and WSDL description of the same data type.
Summary of change:	⌘	Change text description to match current IDL and WSDL of TpCallEventCriteriaResult
Consequences if not approved:	⌘	A contradiction will exist between the IDL and WSDL and the Word document. If no alignment is made, some developers will chose one name, others the other, and interworking problems will arise.

Clauses affected:	⌘	9.2.25

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	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/specs/CR.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.

9.2.25 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
CallEventCriteria	TpCallEventCriteria	The event criteria that were specified by the application.
AssignmentID	TpInt32	The associated assignmentID. This can be used to disable the notification.

←========== FIRST MODIFIED SECTION ==========→

Annex A (normative): OMG IDL Description of Generic Call Control SCF

The OMG IDL representation of this specification is contained in text files gcc_data.idl and gcc_interfaces.idl (contained in archive 291980402IDL.ZIP) which accompany the present document.

```
struct TpCallEventCriteriaResult {
    TpCallEventCriteria CallEventCriteria;
    TpInt32 AssignmentID;
};
```

←========== SECOND MODIFIED SECTION ==========→

Annex B (informative): W3C WSDL Description of Generic Call Control SCF

The W3C WSDL representation of this specification is contained in text files (gcc_data.wsdl, gcc_interfaces.wsdl contained in archive 291980402WSDL.ZIP) which accompanies the present document.

```
<xsd:complexType name="TpCallEventCriteriaResult">
    <xsd:sequence>
        <xsd:element name="AssignmentID" type="osaxsd:TpInt32"/>
        <xsd:element name="CallEventCriteria" type="gcc_dataxsd:TpCallEventCriteria"/>
    </xsd:sequence>
</xsd:complexType>
```

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-04-2 CR 004 ⌘rev - ⌘ Current version: 5.1.0 ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network X

Title:	⌘	Correction to Prepaid Sequence Diagram
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	31/10/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)
Rel-6 (Release 6)

Reason for change:	⌘	The description of the Prepaid sequence diagram in Generic Call Control is incorrect. It indicates that an announcement is played only to party A in a call controlled by a GCC application, when both A and B parties are connected. The announcement will in fact be played to both parties, since there is no means in GCC to separate the two parties in the call. This error has been partially corrected in GCC for Release 5 (N5-020500). This CR completes the changes made in N5-020500.
Summary of change:	⌘	Change the Prepaid sequence diagram to remove indication that B party will not hear announcement.
Consequences if not approved:	⌘	Developers use these sequence diagrams as examples of how OSA/Parlay really behaves. Since they consider that these examples are provided by the real experts, they consider they must be right and should be followed. If we don't correct such errors, we are deliberately misleading developers, and can only expect interoperability problems at later stages.
Clauses affected:	⌘	4.11

Other specs affected: ⌘	Y	N		⌘
		X	Other core specifications	
		X	Test specifications	
		X	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/specs/CR.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.

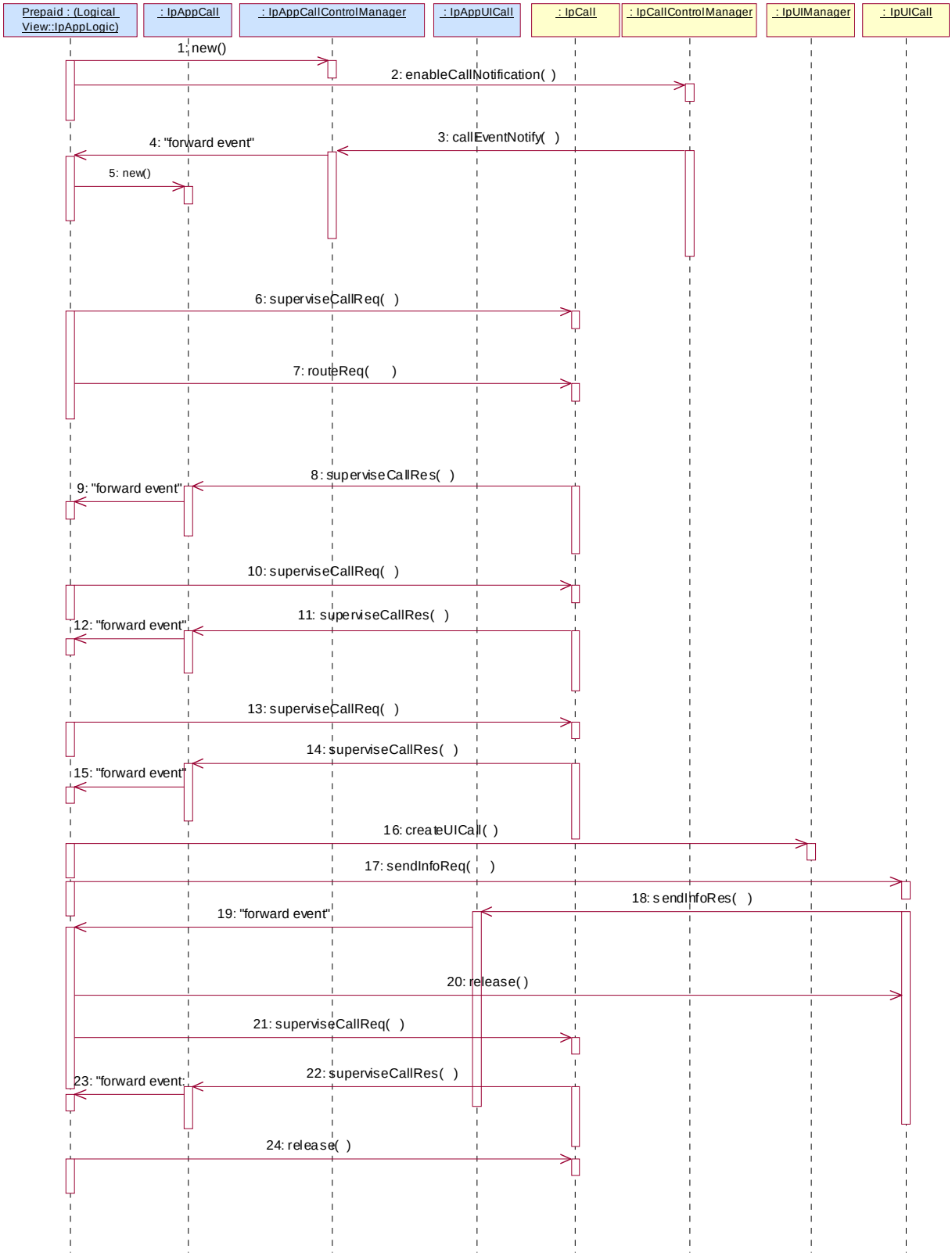
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest 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.

4.11 Prepaid

This sequence shows a Pre-paid application.

The subscriber is using a pre-paid card or credit card to pay for the call. The application each time allows a certain timeslice for the call. After the timeslice, a new timeslice can be started or the application can terminate the call. In the following sequence the end-user will receive an announcement before his final timeslice.



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

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call,

that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

3: The incoming call triggers the Pre-Paid Application (PPA).

4: The message is forwarded to the application.

5: A new object on the application side for the Generic Call object is created

6: The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.

7: Before continuation of the call, PPA sends all charging information, a possible tariff switch time and the call duration supervision period, towards the GW which forwards it to the network.

8: At the end of each supervision period the application is informed and a new period is started.

9: The message is forwarded to the application.

10: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

11: At the end of each supervision period the application is informed and a new period is started.

12: The message is forwarded to the application.

13: The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.

14: When the user is almost out of credit the application is informed.

15: The message is forwarded to the application.

16: The application decides to play an announcement to the parties in this call. A new UICall object is created and associated with the call.

17: An announcement is played informing the user about the near-expiration of his credit limit. ~~The B-subscriber will not hear the announcement.~~

18: When the announcement is completed the application is informed.

19: The message is forwarded to the application.

20: The application releases the UICall object.

21: The user does not terminate so the application terminates the call after the next supervision period.

22: The supervision period ends

23: The event is forwarded to the logic.

24: The application terminates the call. Since the user interaction is already explicitly terminated no userInteractionFaultDetected is sent to the application.

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020873**
Meeting #20, Miami/ FL, USA, 23 – 27 September 2002

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-2** **CR** **003** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Correction of status of GCC methods
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	27/09/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)
Rel-6 (Release 6)

Reason for change:	⌘	There is no requirement in the standard about the necessity to implement all or only some of the methods defined for an interface.
Summary of change:	⌘	Clarify which methods are mandatory and which are optional.
Consequences if not approved:	⌘	Application developers will not know which methods will actually be available.

Clauses affected:	⌘	6 Generic Call Control Service Interface Classes

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	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/specs/CR.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 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.

For the generic call control service, only a subset of the call model defined in clause 4 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 IpCallControlManager 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 IpAppCallControlManager and IpAppCall to provide the callback mechanism.

6.1 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.

This interface shall be implemented by a Generic Call Control SCF. As a minimum requirement either the createCall() method shall be implemented, or the enableCallNotification() and disableCallNotification() methods shall be implemented.

<<Interface>>

IpCallControlManager

createCall (appCall : in IpAppCallRef) : TpCallIdentifier

enableCallNotification (appCallControlManager : in IpAppCallControlManagerRef, eventCriteria : in TpCallEventCriteria) : TpAssignmentID

disableCallNotification (assignmentID : in TpAssignmentID) : void

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

changeCallNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpCallEventCriteria) : void

getCriteria () : TpCallEventCriteriaResultSet

6.1.1 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).

Returns callReference: Specifies the interface reference and sessionID of the call created.

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 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().

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

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.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

6.1.3 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 enableCallNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised. If two callbacks have been registered under this assignment ID both of them will be disabled.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

6.1.4 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.

Returns assignmentID: Specifies the assignmentID assigned by the gateway to this request. This assignmentID can be used to correlate the callOverloadEncountered and callOverloadCeased methods with the request.

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.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

6.1.5 Method changeCallNotification()

This method is used by the application to change the event criteria introduced with enableCallNotification. Any stored criteria associated with the specified assignmentID 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

6.1.6 Method getCriteria()

This method is used by the application to query the event criteria set with enableCallNotification or changeCallNotification.

Returns eventCriteria: Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Parameters

No Parameters were identified for this method

Returns

TpCallEventCriteriaResultSet

Raises

TpCommonExceptions

6.2 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) : void

callEventNotify (callReference : in TpCallIdentifier, eventInfo : in TpCallEventInfo, assignmentID : in TpAssignmentID) : IpAppCallRef

callNotificationInterrupted () : void

callNotificationContinued () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

6.2.1 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.

6.2.2 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_CALL_MONITOR_MODE_INTERRUPT, 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_CALL_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.

Returns appCall: Specifies a reference to the application interface which implements the callback interface for the new call. If the application has previously explicitly passed a reference to the IpAppCall interface using a setCallback() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallback().

This parameterwill be null if the notification is in NOTIFY mode.

Parameters

`callReference : in TpCallIdentifier`

Specifies the reference to the call interface to which the notification relates. If the notification is in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking callEventNotify may populate this parameter as it chooses.

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

Specifies the assignment id which was returned by the enableCallNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

Returns

IpAppCallRef

6.2.3 Method callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporarily 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

6.2.4 Method callNotificationContinued()

This method indicates to the application that event notifications will again be possible.

Parameters

No Parameters were identified for this method

6.2.5 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 address range for within which the overload has been encountered.

6.2.6 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

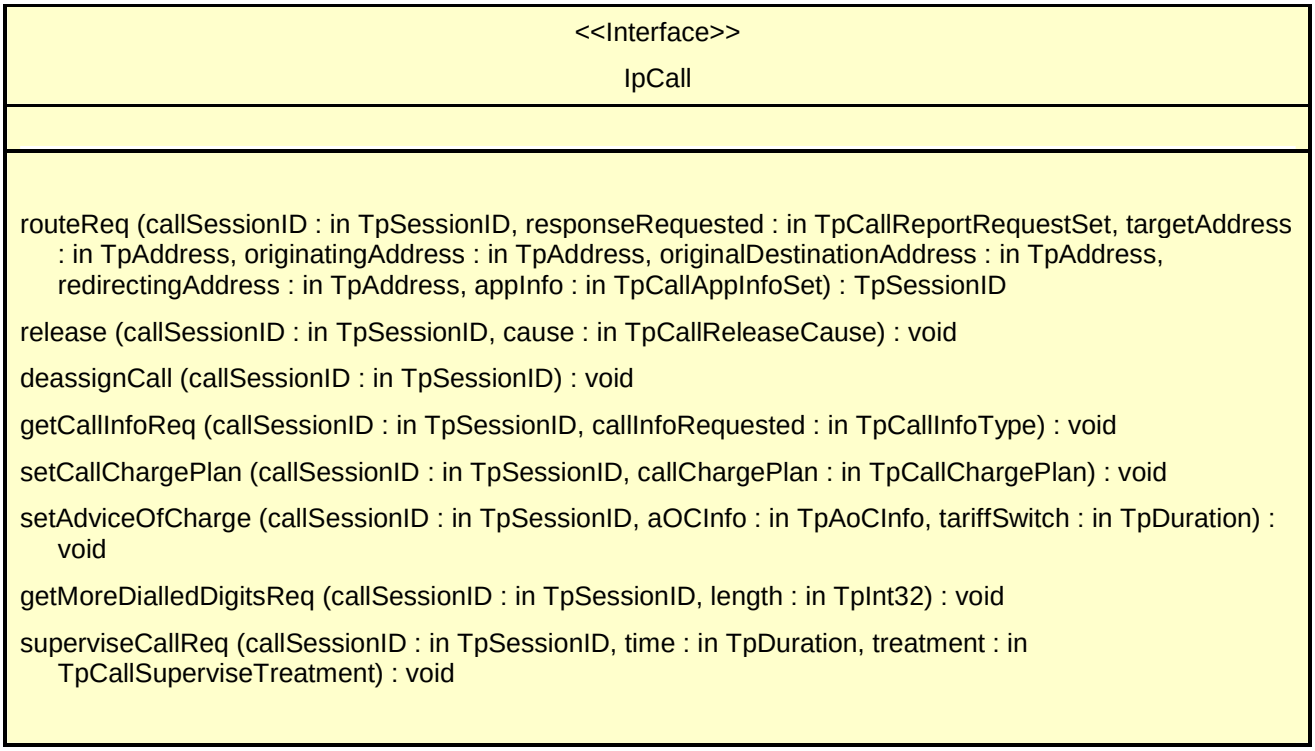
Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the address range for within which the overload has been ceased

6.3 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.

This interface shall be implemented by a Generic Call Control SCF. As a minimum requirement, the routeReq (), release() and deassignCall() methods shall be implemented.



6.3.1 Method routeReq()

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

Note that in case of routeReq() it is recommended to request for 'successful' (e.g. 'answer' event) and 'failure' events at invocation, because those are needed for the application to keep track of the state of the call.

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.

Returns callLegSessionID: 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.

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).

Returns

TpSessionID

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

6.3.2 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

6.3.3 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

6.3.4 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.

A report 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

TpCommonExceptions, P_INVALID_SESSION_ID

6.3.5 Method setCallChargePlan()

Set an operator specific charge plan for the call.

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

6.3.6 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

6.3.7 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

6.3.8 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

6.4 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) : void routeErr (callSessionID : in TpSessionID, errorIndication : in TpCallError, callLegSessionID : in TpSessionID) : void getCallInfoRes (callSessionID : in TpSessionID, callInfoReport : in TpCallInfoReport) : void getCallInfoErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void superviseCallRes (callSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedTime : in TpDuration) : void superviseCallErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void callFaultDetected (callSessionID : in TpSessionID, fault : in TpCallFault) : void getMoreDialledDigitsRes (callSessionID : in TpSessionID, digits : in TpString) : void getMoreDialledDigitsErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void callEnded (callSessionID : in TpSessionID, report : in TpCallEndedReport) : void

6.4.1 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_CALL_MONITOR_MODE_INTERRUPT,

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 sessionID returned at the routeReq() and can be used to correlate the response with the request.

6.4.2 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.

6.4.3 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.

6.4.4 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.

6.4.5 Method superviseCallRes()

This asynchronous method reports a call supervision event to the application when it has indicated its 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).

6.4.6 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.

6.4.7 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.

6.4.8 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.

6.4.9 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.

6.4.10 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.