3GPP TSG CN Plenary Meeting #19 **NP-030031**
12- 14 March 2003, Birmingham, UK

Source: **CN5 (OSA)**

Title: **Rel-5 CRs 29.198-04-3 OSA API Part 4: Call control;**
Sub-part 3: Multi-Party Call Control SCF

Agenda item: **8.2**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
NP-030031	29.198-04-3	007	-	Rel-5	Correction of status of MPCC methods	F	5.1.0	N5-020874	OSA2
NP-030031	29.198-04-3	008	-	Rel-5	Inconsistent description of use of secondary callback	F	5.1.0	N5-021038	OSA2

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3** **CR** **007** ⌘ **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 MPCC methods
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	27/09/2002
Category:	⌘	**F**
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 Multi Party 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 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.

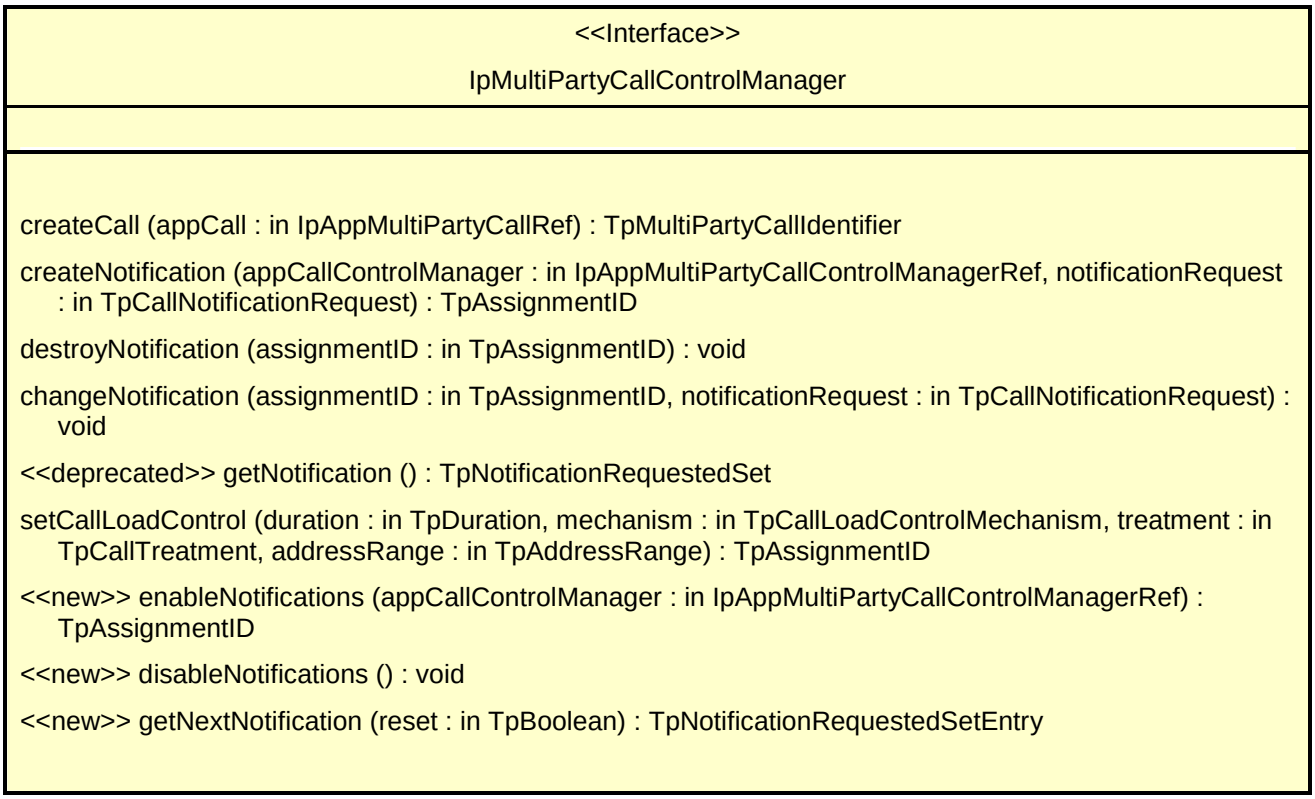
The Multi-party Call Control Service is represented by the IpMultiPartyCallControlManager, IpMultiPartyCall, IpCallLeg interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way,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

6.1 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, or the enableNotifications() and disableNotifications() methods shall be implemented.



6.1.1 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

6.1.2 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 or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap when it leads to more than one application controlling the call or session at the same point in time during call or session processing.

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 createNotification 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

6.1.3 Method destroyNotification()

This method is used by the application to disable call notifications. This method only applies to notifications created with createNotification().

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the multi party call control manager interface when the previous createNotification() 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 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 multi party 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

6.1.5 Method <<deprecated>> getNotification()

This method is deprecated and replaced by getNextNotification(). It will be removed in a later release.

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. An empty set is returned when no notifications exist.

Parameters

No Parameters were identified for this method

Returns

TpNotificationRequestedSet

Raises

TpCommonExceptions

6.1.6 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

6.1.7 Method <<new>> enableNotifications()

This method is used to indicate that the application is able to receive notifications which are provisioned from within the network (i.e. these notifications are NOT set using createNotification() but via, for instance, a network management system). If notifications provisioned for this application are created or changed, the application is unaware of this until the notification is reported.

If the same application requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. This means that the callback will only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).

When this method is used, it is still possible to use createNotification() for service provider provisioned notifications on the same interface as long as the criteria in the network and provided by createNotification() do not overlap. However, it is NOT recommended to use both mechanisms on the same service manager.

The methods changeNotification(), getNotification(), and destroyNotification() do not apply to notifications provisoned in the network and enabled using enableNotifications(). These only apply to notifications created using createNotification().

Returns assignmentID: Specifies the ID assigned by the manager interface for this operation. This ID is contained in any reportNotification() that relates to notifications provisioned from within the networkRepeated calls to enableNotifications() return the same assignment ID.

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.

Returns

TpAssignmentID

Raises

TpCommonExceptions

6.1.8 Method <<new>> disableNotifications()

This method is used to indicate that the application is not able to receive notifications for which the provisioning has been done from within the network. (i.e. these notifications that are NOT set using createNotification() but via, for instance, a network management system). After this method is called, no such notifications are reported anymore.

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

6.1.9 Method <<new>> getNextNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification. Since a lot of data can potentially be returned (which might cause problem in the middleware), this method must be used in an iterative way. Each method invocation may return part of the total set of notifications if the set is too large to return it at once. The reset parameter permits the application to indicate whether an invocation to getNextNotification is requesting more notifications from the total set of notifications or is requesting that the total set of notifications shall be returned from the beginning.

Returns notificationRequestedSetEntry: The set of notifications and an indication whether all off the notifications have been obtained or if more notifications are available that have not yet been obtained by the application. If no notifications exist, an empty set is returned and the final indication shall be set to TRUE.

Note that the (maximum) number of items provided to the application is determined by the gateway.

Parameters

reset : in TpBoolean

TRUE: indicates that the application is intended to obtain the set of notifications starting at the beginning.

FALSE: indicates that the application requests the next set of notifications that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.

The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in TpNotificationRequestedSetEntry, for the next call to this method reset shall be set to TRUE. P_TASK_REFUSED may be thrown if these conditions are not met.

Returns

TpNotificationRequestedSetEntry

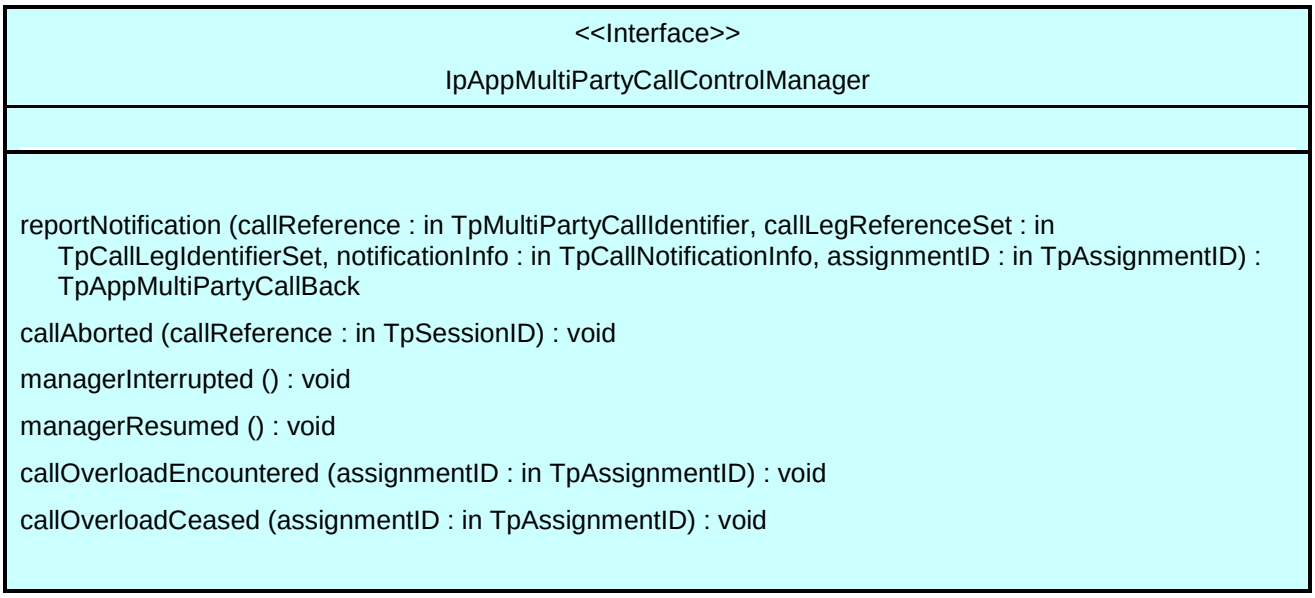
Raises

TpCommonExceptions

6.2 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.



6.2.1 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

6.2.2 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.

6.2.3 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

6.2.4 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

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 addressrange 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

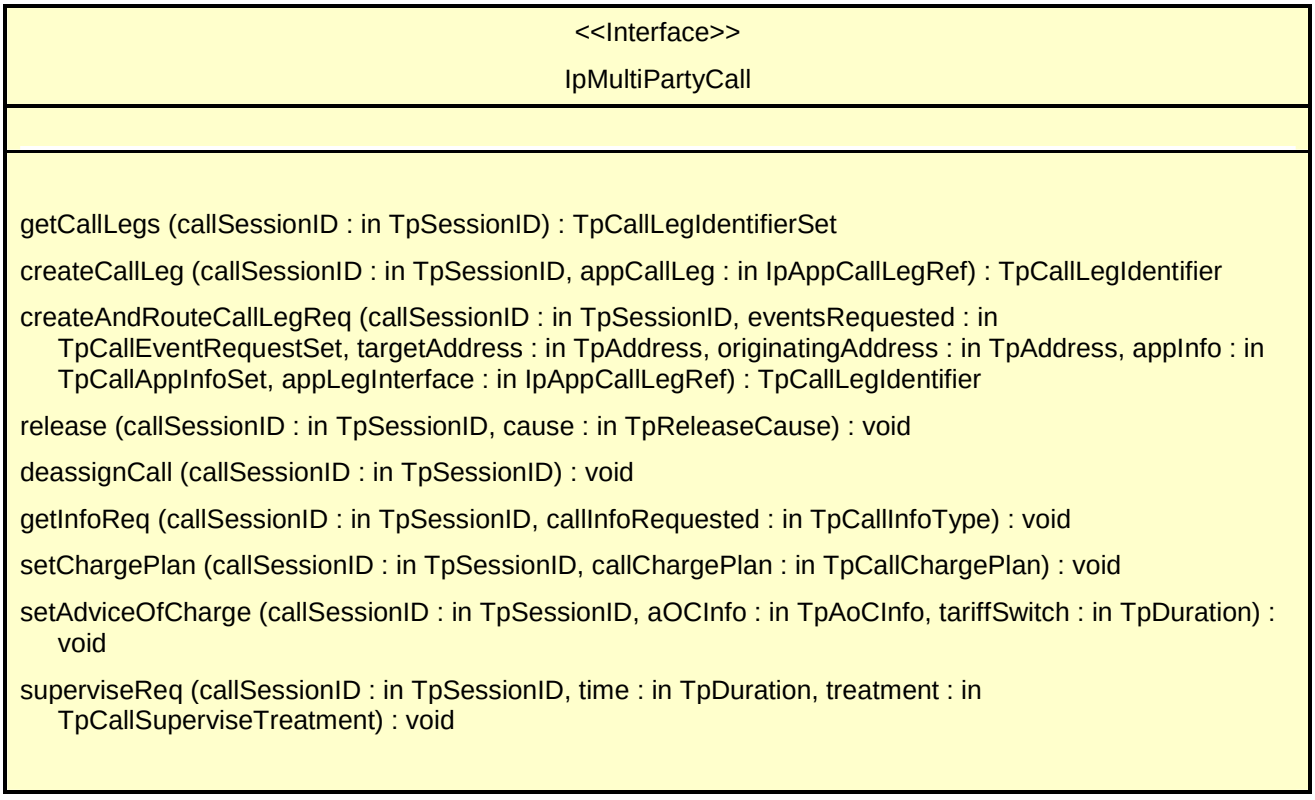
Specifies the assignmentID corresponding to the associated setCallLoadControl. This implies the addressrange for within which the overload has been ceased

6.3 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.



6.3.1 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

6.3.2 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

6.3.3 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.

Note that for application initiated calls in some networks the result of the first createAndRouteCallLegReq() has to be received before the next createAndRouteCallLegReq() can be invoked. The Service Property P_PARALLEL_INITIAL_ROUTING_REQUESTS (see section 7.5) indicates how a specific implementation handles the initial createAndRouteCallLegReq(). This method shall throw P_TASK_REFUSED if an application is not allowed to use parallel routing requests.

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

6.3.4 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

6.3.5 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.

When this method is invoked, all outstanding supervision requests will be cancelled.

Parameters

callSessionID : in TpSessionID

Specifies the call session ID of the call.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.3.6 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

6.3.7 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

6.3.8 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

6.3.9 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. Measurement will start as soon as the call is connected in the network, e.g. answered by the B-party or the user-interaction system.

treatment : in TpCallSuperviseTreatment

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

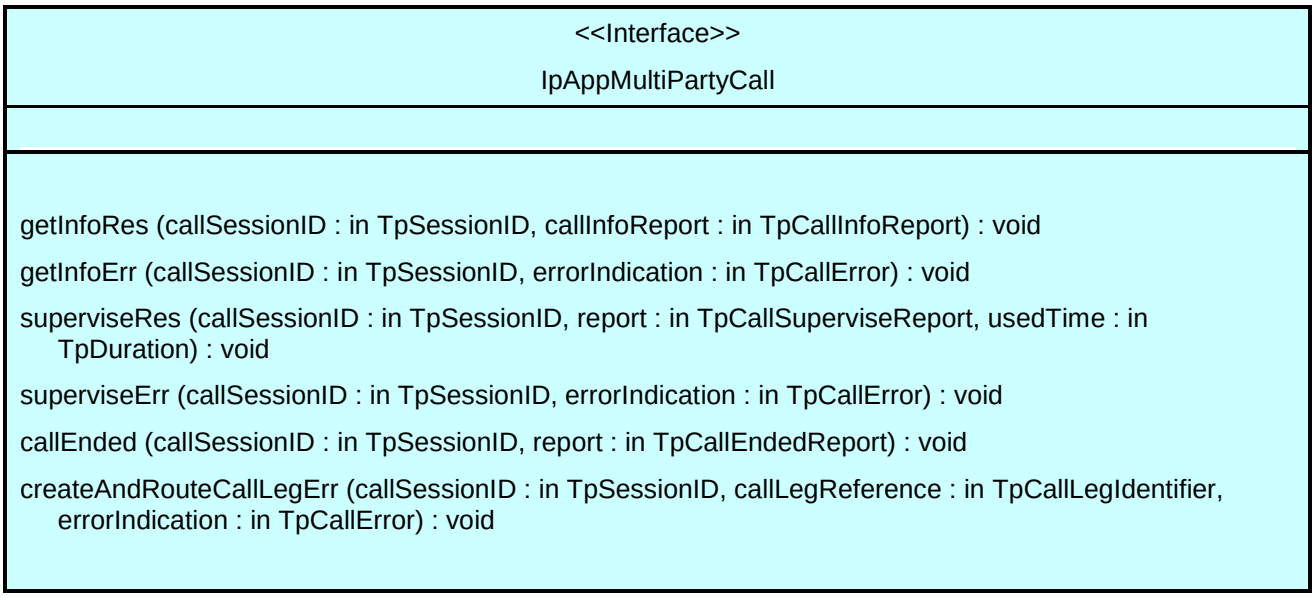
Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.4 Interface Class 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.



6.4.1 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.

6.4.2 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.

6.4.3 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).

6.4.4 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.

6.4.5 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.

6.4.6 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.

6.5 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.

<<Interface>> IpCallLeg
routeReq (callLegSessionID : in TpSessionID, targetAddress : in TpAddress, originatingAddress : in TpAddress, appInfo : in TpCallAppInfoSet, connectionProperties : in TpCallLegConnectionProperties) : void eventReportReq (callLegSessionID : in TpSessionID, eventsRequested : in TpCallEventRequestSet) : void release (callLegSessionID : in TpSessionID, cause : in TpReleaseCause) : void getInfoReq (callLegSessionID : in TpSessionID, callLegInfoRequested : in TpCallLegInfoType) : void getCall (callLegSessionID : in TpSessionID) : TpMultiPartyCallIdentifier attachMediaReq (callLegSessionID : in TpSessionID) : void detachMediaReq (callLegSessionID : in TpSessionID) : void getCurrentDestinationAddress (callLegSessionID : in TpSessionID) : TpAddress continueProcessing (callLegSessionID : in TpSessionID) : void setChargePlan (callLegSessionID : in TpSessionID, callChargePlan : in TpCallChargePlan) : void setAdviceOfCharge (callLegSessionID : in TpSessionID, aOCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : void superviseReq (callLegSessionID : in TpSessionID, time : in TpDuration, treatment : in TpCallLegSuperviseTreatment) : void deassign (callLegSessionID : in TpSessionID) : void

6.5.1 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.

Note that for application initiated calls in some networks the result of the first routeReq() has to be received before the next routeReq() can be invoked. The Service Property P_PARALLEL_INITIAL_ROUTING_REQUESTS (see section 7.5) indicates how a specific implementation handles the initial routeReq().This method shall throw P_TASK_REFUSED if an application is not allowed to use parallel routing requests.

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

6.5.2 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

6.5.3 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

6.5.4 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

6.5.5 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

6.5.6 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.

In case this method is invoked while there is still a request to detach the Media pending, the exception "P_TASK_REFUSED" will be raised.

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

6.5.7 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.

In case this method is invoked while there is still a request to attach the Media pending, the exception "P_TASK_REFUSED" will be raised.

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

6.5.8 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

6.5.9 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.

In case the operation is invoked and call leg processing is not interrupted the exception P_INVALID_NETWORK_STATE will be raised.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

6.5.10 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

6.5.11 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

6.5.12 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. Measurement will start as soon as the callLeg is connected in the network.

treatment : in TpCallLegSuperviseTreatment

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

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.13 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.

When this method is invoked, all outstanding supervision requests will be cancelled.

Parameters

callLegSessionID : in TpSessionID

Specifies the call leg session ID of the call leg.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.6 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

6.6.1 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.

6.6.2 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.

6.6.3 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.

6.6.4 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.

6.6.5 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.

6.6.6 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.

6.6.7 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.

6.6.8 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.

6.6.9 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.

6.6.10 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).

6.6.11 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.

6.6.12 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-v5

CHANGE REQUEST

⌘ 29.198-04-3 CR 008 ⌘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: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network X

Title:	⌘	Inconsistent description of use of secondary callback
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	10/10/2002
Category:	⌘	**F**
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)

Reason for change:	⌘	OSA Specification describes use of secondary callback interface inconsistently between the different parts which confuses application developers.
Summary of change:	⌘	Describe that most recent call back will be used as the callback interface. Only if this one does not work, the initially provided callback interface is used.
Consequences if not approved:	⌘	Interoperability problems.

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

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

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

Introduction

The OSA Specifications contains the following descriptions about the use of a secondary callback interface:

Part 1:

7.12 Notification Handling

It is possible to recreate a (set of) notification(s) or re-register for notifications. This is only useful when providing a different callback interface reference. In this case, the last provided interface is used for reporting notifications. ***The earlier provided callback interface is used as "backup" interface (this can be the one provided with setCallback() or setCallbackWithSessionID() if NULL was provided initially). Notifications are reported on this interface when calls to the most recent provided callback interface fail (object providing the interface is crashed or overloaded).*** When re-creating or re-registering, the same assignment ID is returned.

Part 4-2:

6.1.2 Method enableCallNotification()

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().

Part 4-3:

6.1.2 Method createNotification()

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 createNotification 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().

6.1.7 Method <<new>> enableNotifications()

If the same application requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them***), the second callback will be treated as an additional callback. This means that the callback will only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).***

Part 4-4:

6.1.1 Method createMediaNotification()

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 createMediaNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the one that has been registered by setCallback().

Part 5:

8.1.3 Method createNotification()

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. This means that the callback will only be used in case when the first callback specified by the application is unable to handle the reportNotification (e.g., due to overload or failure).***

8.1.7 Method <<new>> enableNotifications()

If the same application requests to enable notifications for a second time with a different IpAppUIManager reference (i.e. without first disabling them***), the second callback will be treated as an additional callback. This means that the callback wil only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).***

Part 8:

8.4.1 Method <<deprecated>> createNotification()

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 createNotification 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().

8.4.5 Method <<new>> enableNotifications()

If the same application requests to enable notifications for a second time with a different IpAppDataSessionControlManager reference (i.e. without first disabling them), ***the second callback will be treated as an additional callback. This means that the callback wil only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).***

8.4.8 Method <<new>> createNotifications()

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 createNotification 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().

Part 11:

8.1.1 Method createNotification()

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().

8.1.7 Method <<new>> enableNotifications()

If the same application requests to enable notifications for a second time with a different IpAppAccountManager reference (i.e. without first disabling them), ***the second callback will be treated as an additional callback. This means that the callback will only be used in cases when the first callback specified by the application is unable to handle the reportNotification (e.g. due to overload or failure).***

Solution

The intended use of the 2nd callback interface is as described in part 1, therfore the changes to the following method descriptions are proposed:

- Part 4-3, method enableNotifications()
- Part 5, method createNotification()
- Part 5, method enableNotifications()
- Part 8, method enableNotifications()
- Part 11, method enableNotifications()

This contribution proposes the changes for Part 4-3.

Proposed Changes

6.1.7 Method <<new>> enableNotifications()

This method is used to indicate that the application is able to receive notifications which are provisioned from within the network (i.e. these notifications are NOT set using createNotification() but via, for instance, a network management system). If notifications provisioned for this application are created or changed, the application is unaware of this until the notification is reported.

If the same application requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.~~This means that the callback will only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).~~

When this method is used, it is still possible to use createNotification() for service provider provisioned notifications on the same interface as long as the criteria in the network and provided by createNotification() do not overlap. However, it is NOT recommended to use both mechanisms on the same service manager.

The methods changeNotification(), getNotification(), and destroyNotification() do not apply to notifications provisoned in the network and enabled using enableNotifications(). These only apply to notifications created using createNotification().

Returns assignmentID: Specifies the ID assigned by the manager interface for this operation. This ID is contained in any reportNotification() that relates to notifications provisioned from within the networkRepeated calls to enableNotifications() return the same assignment ID.

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.

Returns

TpAssignmentID

Raises

TpCommonExceptions