3GPP TSG CN Plenary Meeting #26
08-10 December 2004, Athens, GREECE

NP-040483

Source:	**CN5 (OSA)**
Title:	**5 Rel-4/5/6 CR 29.198-04 OSA API Part 4: Call control**
Agenda item:	**7.10 (OSA Enhancements [OSA1])**
Document for:	**APPROVAL**

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
NP-040483	29.198-04	071	--	Rel-4	Correct Behaviour of CallBack sequence and timing	F	4.10.0	N5-040718	OSA1
NP-040483	29.198-04-2	025	--	Rel-5	Correct Behaviour of CallBack sequence and timing	A	5.8.0	N5-040719	OSA1
NP-040483	29.198-04-2	026	--	Rel-6	Correct Behaviour of CallBack sequence and timing	A	6.2.0	N5-040721	OSA1
NP-040483	29.198-04-3	032	--	Rel-5	Correct Behaviour of CallBack sequence and timing	A	5.8.0	N5-040720	OSA1
NP-040483	29.198-04-3	033	--	Rel-6	Correct Behaviour of CallBack sequence and timing	A	6.3.0	N5-040722	OSA1

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04**	**CR**	**071**	⌘ **rev**	**-**	⌘	Current version:	**4.10.0** ⌘

*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: ⌘	Correct Behaviour of CallBack sequence and timing		
Source: ⌘	CN5 AePONA (Eamonn Murray)		
Work item code: ⌘	OSA1	***Date:*** ⌘	05/11/2004
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: ⌘	Misunderstandings in how to treat and use the callback functionality supported in call control services has been reported from the second OSA/Parlay PLUGTEST event. In particular the sequence and timing of specifying callbacks has been subject to different interpretations amongst vendors of applications and services. This was recognised as a major interoperability problem at the second OSA/Parlay Interoperability test. As a result, CRs to improve the description of callback behaviour were introduced during CN5#27 as contributions N5-040338 through N5-040342. However the resulting specification text that has been produced, retains significant ambiguities regarding the use of the callback functionality, such that interoperability between application and service implementations, using the callback features, cannot be clearly understood. It is therefore recommended to further clarify the description of the use of the callback features such that a clear and common understanding is possible for vendors of applications and services.
Summary of change: ⌘	Correctly define the use of callback features through text corrections to existing method semantics and correction to existing sequence diagram.
Consequences if not approved: ⌘	Interoperability cannot be supported, as the existing specification shall remain open to mutliple, differing, interpretations.

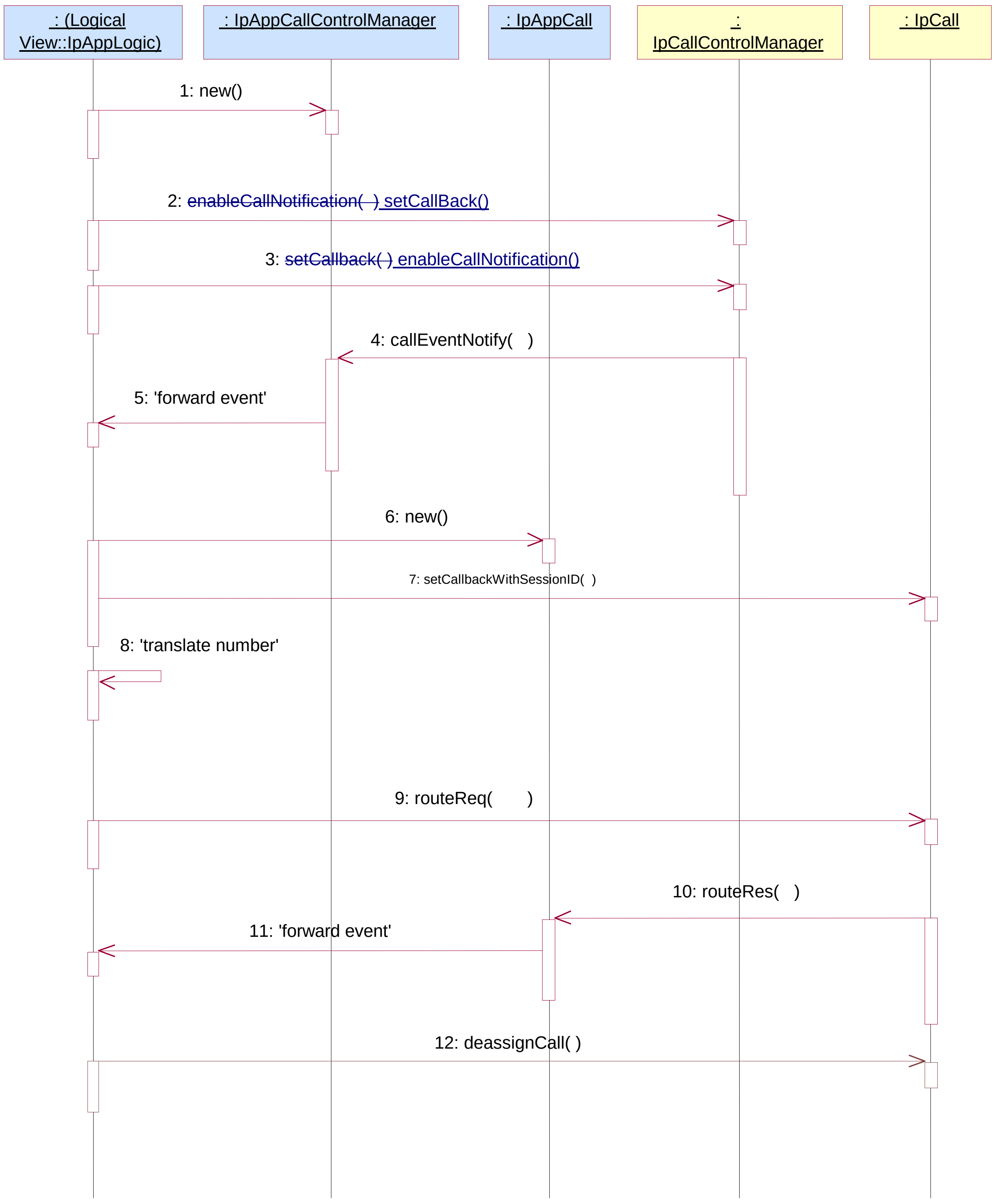
Clauses affected: ⌘	6.1.6, 6.3.1, 6.3.2, 7.3.1, 7.3.2				
	Y	**N**			
Other specs affected: ⌘	X		Other core specifications	⌘	Rel-5/6: 29.198-04-2, 29.198-04-3
		X	Test specifications		
		X	O&M Specifications		
Other comments: ⌘	Rel 5 Mirror CRs in N5-040719 and N5-040720. Rel 6 Mirror CRs in N5-040721 and N5-040722				

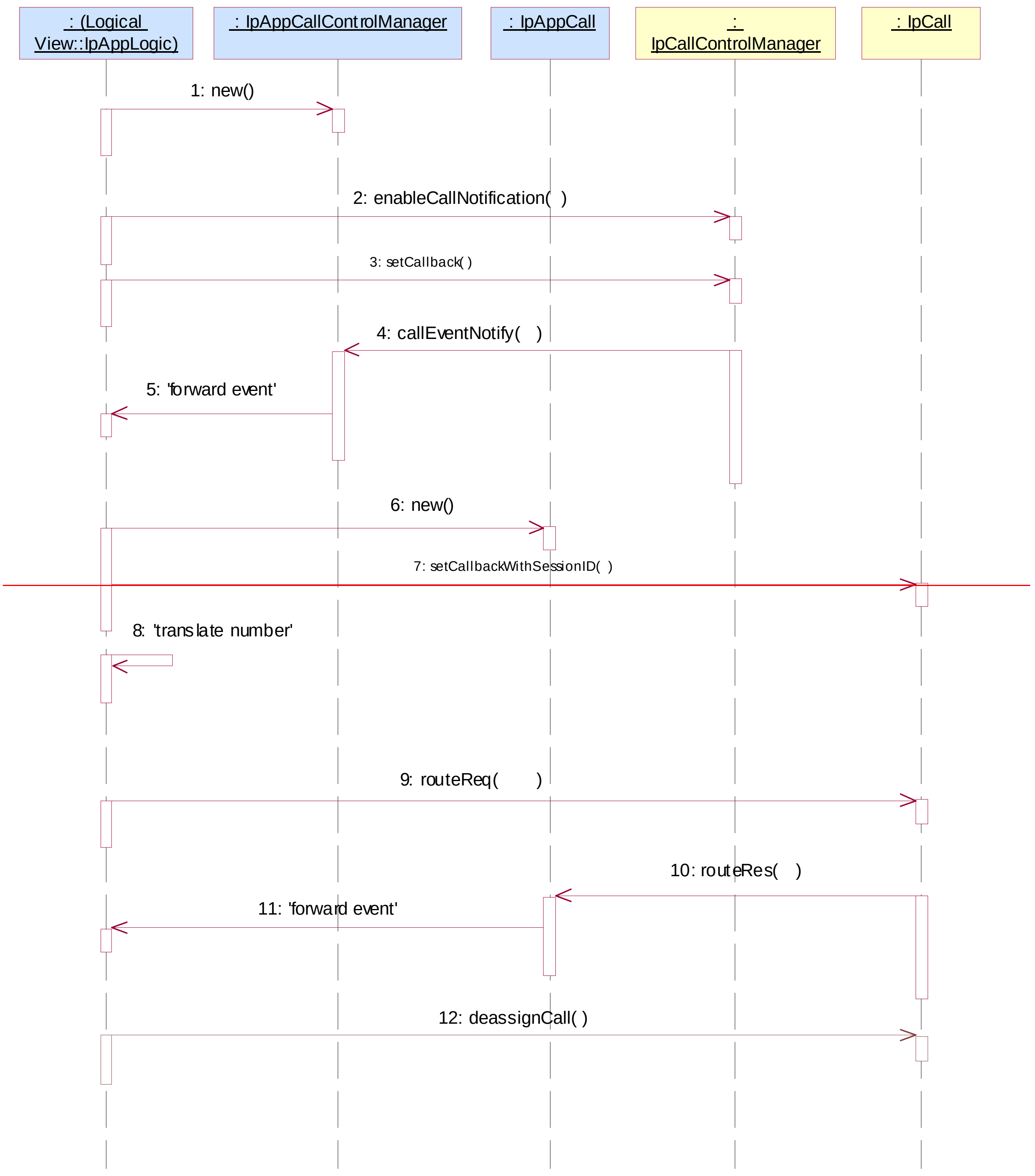
Change in Clause 6.1.6

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.

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.


: (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpCallControlManager
: IpCall
1: new()
2: ~~enableCallNotification( )~~ setCallBack()
3: ~~setCallback( )~~ enableCallNotification()
4: callEventNotify( )
5: 'forward event'
6: new()
7: setCallbackWithSessionID( )
8: 'translate number'
9: routeReq( )
10: routeRes( )
11: 'forward event'
12: deassignCall( )



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.~~ 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.

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.~~ 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 3, 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.

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.

End of Change in Clause 6.1.6

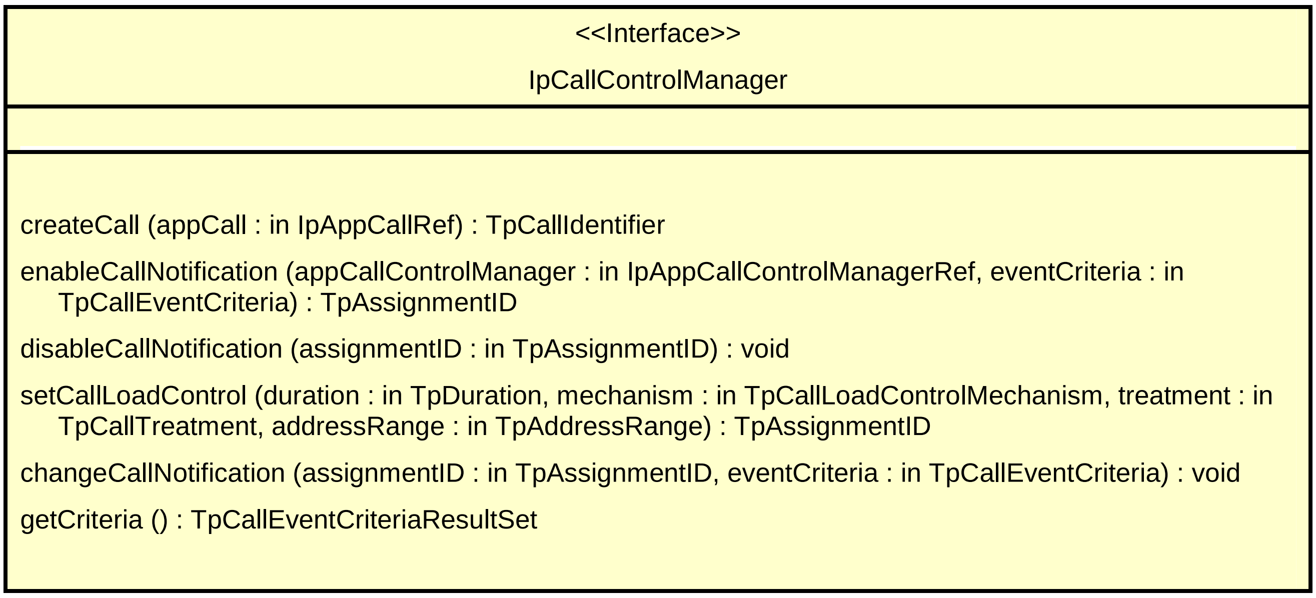
Change in Clause 6.3.1

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.


<<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

Method
createCall()

This method is used to create a new call object.

Call back reference:

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~~ shall invoke setCallback() prior to createCall 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.

Set of the callback reference:

The call back reference can be registered either in a) enableCallNotification() or b) explicitly with a separate setCallback() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the enableCallNotification() with explicit immediate registration (no "Null" value) of call back reference may be the preferred method.

Case b:

The enableCallNotfication() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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(). See example in 6.1.6

Set additional callback reference:

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. See examples in 6.1.1.

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 previously 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`

End of Change in Clause 6.3.1

Change in Clause 6.3.2

6.3.2 Interface Class IpAppCallControlManager

Inherits from: IpInterface

The generic call control manager application interface provides the application call control management functions to the generic call control service.

<<Interface>> IpAppCallControlManager
callAborted (callReference : in TpSessionID) : 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

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).

Set of the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates. However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode.

When callEventNotify() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. routeReq() is performed until an IpAppCall has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

The call back reference can be registered either in a) callEventNotify() or b) explicitly with a setCallbackWithSessionID() method e.g. depending on how the application provides its call reference.

Case a:

From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.

Case b:

The callEventNotify() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted.

In case the callEventNotify() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID(). See example in 6.1.6

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 setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

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

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`

End of Change in Clause 6.3.2

Change in Clause 7.3.1

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.

<<Interface>> IpMultiPartyCallControlManager
createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier createNotification (appCallControlManager : in IpAppMultiPartyCallControlManagerRef, notificationRequest : in TpCallNotificationRequest) : TpAssignmentID destroyNotification (assignmentID : in TpAssignmentID) : void changeNotification (assignmentID : in TpAssignmentID, notificationRequest : in TpCallNotificationRequest) : void getNotification () : TpNotificationRequestedSet setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange) : TpAssignmentID

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~~ shall invoke setCallback() prior to createCall() 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.

Set of the callback reference:

The call back reference can be registered either in a) createNotication() or b) explicitly with a setCallback() method e.g. depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the createNotification() with explicit registration may be the preferred method.

Case b:

The createNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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().

Set additional Call back:

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 previously 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`

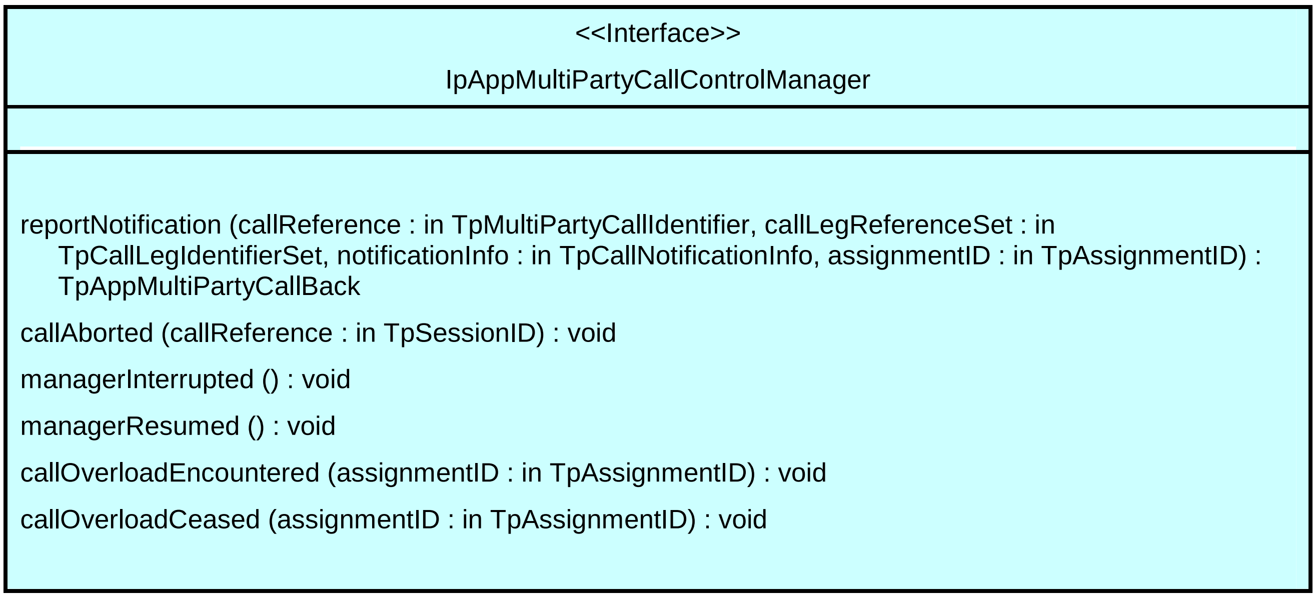
End of Change in Clause 7.3.1

Change in Clause 7.3.2

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.


<<Interface>>

IpAppMultiPartyCallControlManager

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

callAborted (callReference : in TpSessionID) : void

managerInterrupted () : void

managerResumed () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

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.

Set of the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates. However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode.

When reportNotification() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. createAndRouteCallLegReq() is performed until the callback interface for the new call and/or new call leg has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation, or via the return of the reportNotification() method.

The call back reference can be registered either in a) reportNotification() or b) explicitly with a setCallbackWithSessionID() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the reportNotification() with explicit pass of registration may be the preferred method.

Case b:

The reportNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID(). If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted

In case reportNotification() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID().

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 setCallbackWithSessionID() invocation, this parameter may be set to P_APP_CALLBACK_UNDEFINED, or if supplied must be the same as that provided during the setCallbackWithSessionID().

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

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`

End of Change in Clause 7.3.2

End Of Document

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-2**	**CR**	**025**	⌘**rev**	**-**	⌘	Current version:	**5.8.0** ⌘

*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:	⌘	Correct Behaviour of CallBack sequence and timing		
Source:	⌘	CN5 AePONA (Eamonn Murray)		
Work item code:	⌘	OSA1	***Date:*** ⌘	05/11/2004
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: ⌘ Misunderstandings in how to treat and use the callback functionality supported in call control services has been reported from the second OSA/Parlay PLUGTEST event. In particular the sequence and timing of specifying callbacks has been subject to different interpretations amongst vendors of applications and services. This was recognised as a major interoperability problem at the second OSA/Parlay Interoperability test.

As a result, CRs to improve the description of callback behaviour were introduced during CN5#27 as contributions N5-040338 through N5-040342. However the resulting specification text that has been produced, retains significant ambiguities regarding the use of the callback functionality, such that interoperability between application and service implementations, using the callback features, cannot be clearly understood.

It is therefore recommended to further clarify the description of the use of the callback features such that a clear and common understanding is possible for vendors of applications and services.

Summary of change: ⌘ Correctly define the use of callback features through text corrections to existing method semantics and correction to existing sequence diagram.

Consequences if not approved: ⌘ Interoperability cannot be supported, as the existing specification shall remain open to mutliple, differing, interpretations.

Clauses affected: ⌘ 4.6, 6.1, 6.2,

		Y	N			
Other specs affected:	⌘	X		Other core specifications	⌘	Rel-6: 29.198-04-2
			X	Test specifications		
			X	O&M Specifications		

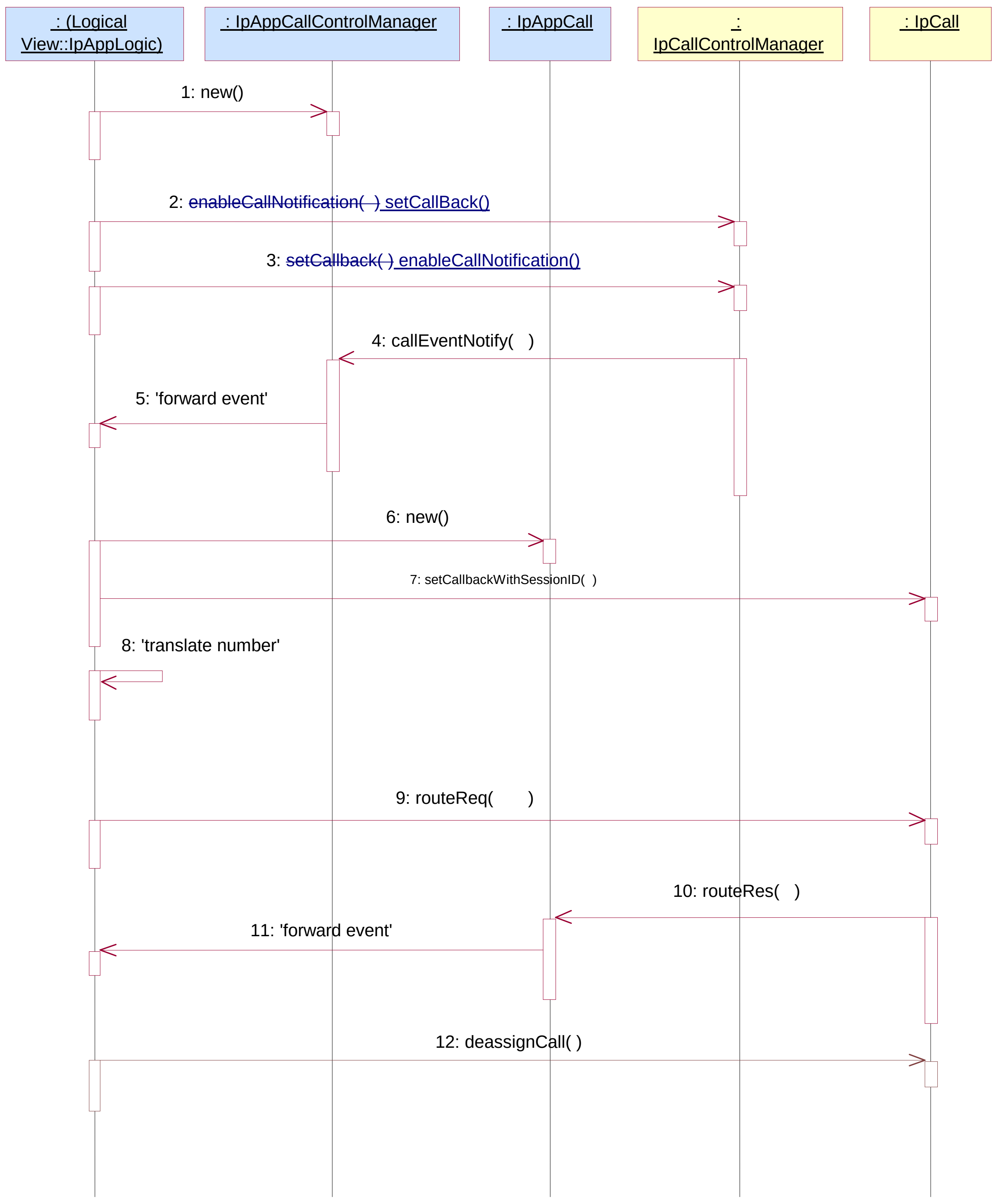
Other comments: ⌘ Rel-5 Mirror CR to N5-040718 Rel 4 CR 29.198-04

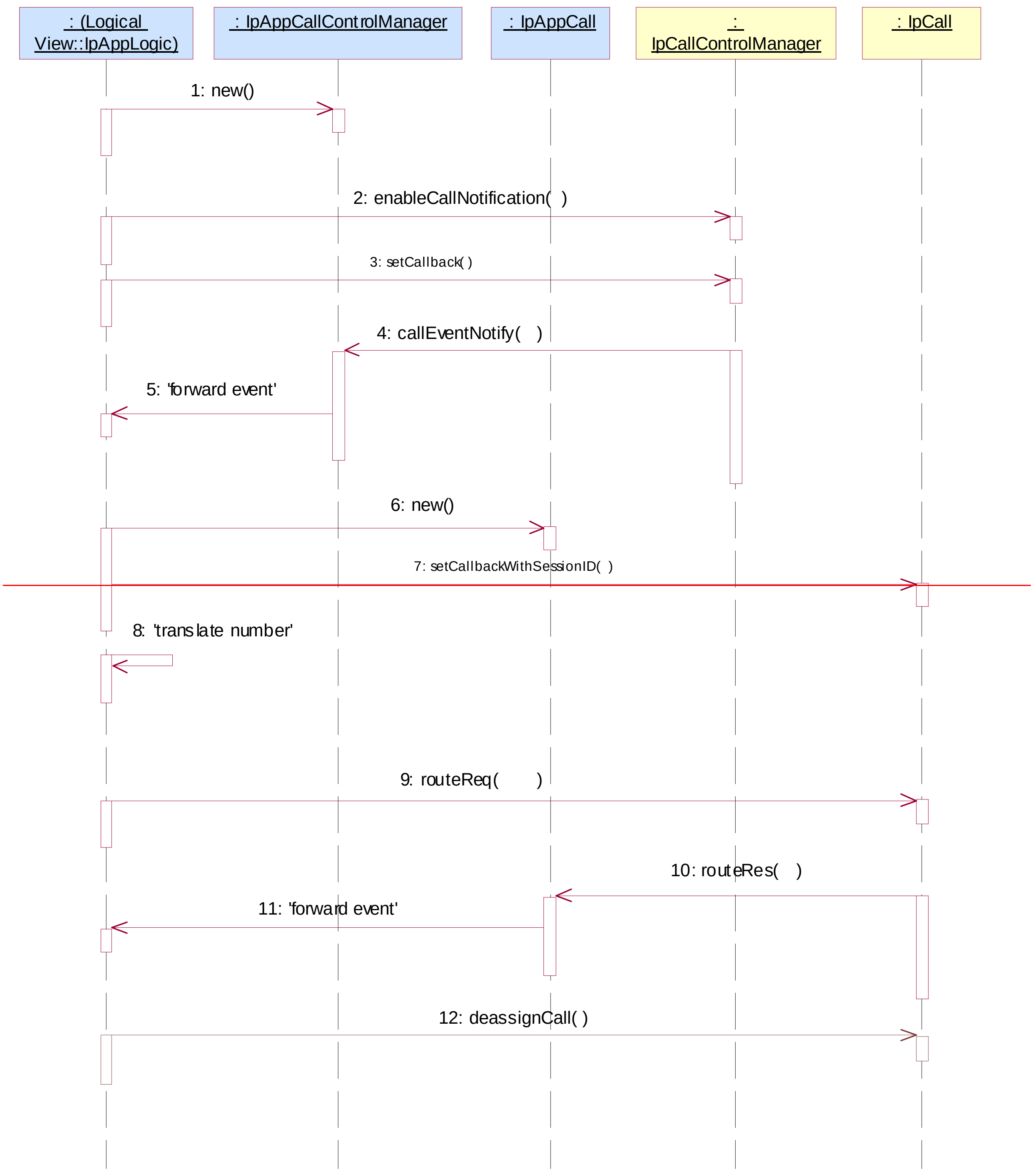
Change in Clause 4.6

4.6 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.

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.


: (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpCallControlManager
: IpCall
1: new()
2: ~~enableCallNotification( )~~ setCallBack()
3: ~~setCallback( )~~ enableCallNotification()
4: callEventNotify( )
5: 'forward event'
6: new()
7: setCallbackWithSessionID( )
8: 'translate number'
9: routeReq( )
10: routeRes( )
11: 'forward event'
12: deassignCall( )



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.~~ 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.

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.~~ 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 3, 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.

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.

End of Change in Clause 4.6

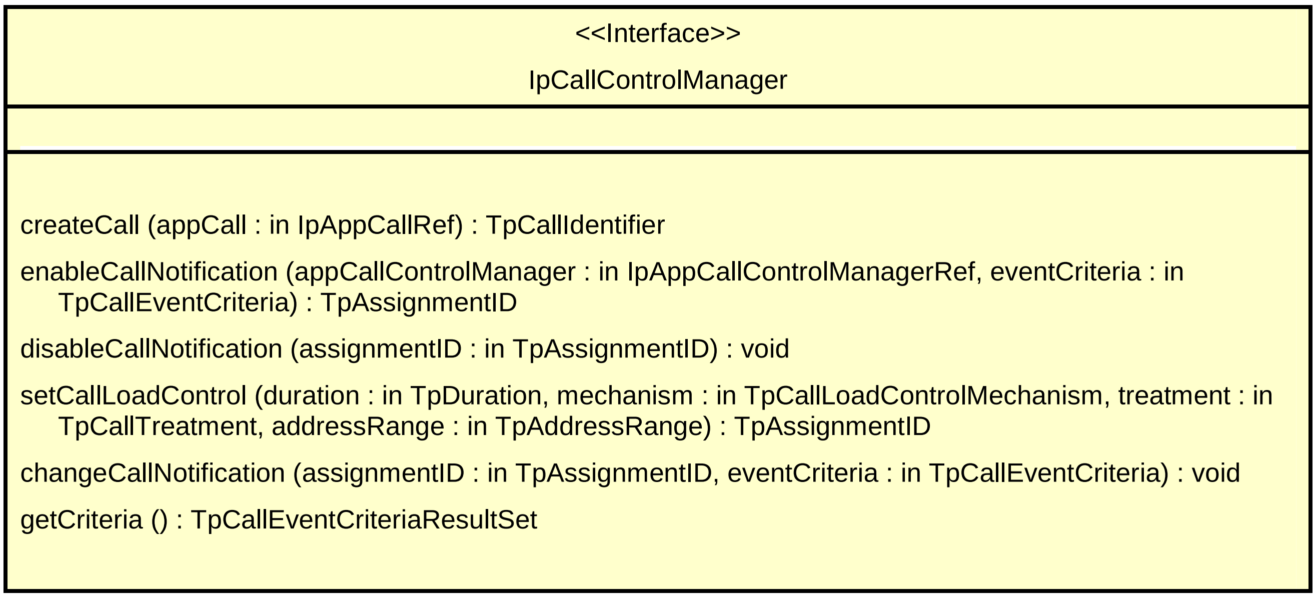
Change in Clause 6.1

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.

Call back reference:

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~~ shall invoke setCallback() prior to createCall 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.

Setting the callback reference:

The call back reference can be registered either in a) enableCallNotification() or b) explicitly with a separate setCallback() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the enableCallNotification() with explicit immediate registration (no "Null" value) of call back reference may be the preferred method.

Case b:

The enableCallNotfication() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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(). See example in 6.1.6

Set additional callback:

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. See examples in 6.1.1.

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 previously 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`

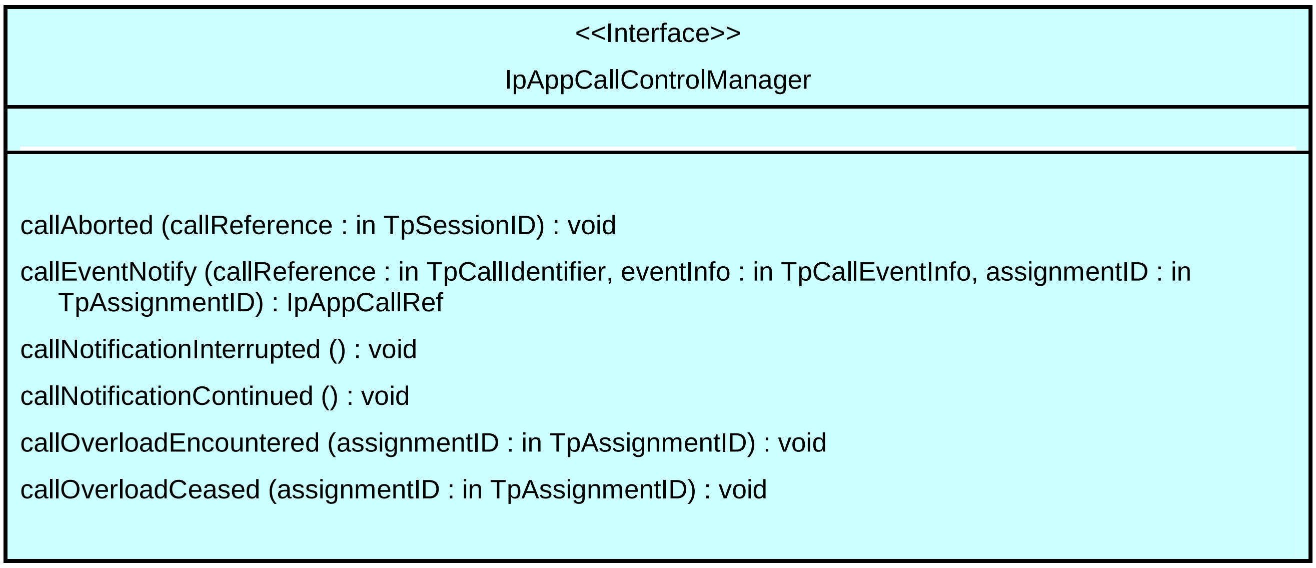
End of Change in Clause 6.1

Change in Clause 6.2

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).

Setting the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates. However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode.

When callEventNotify() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. routeReq() is performed until an IpAppCall has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

The call back reference can be registered either in a) callEventNotify() or b) explicitly with a setCallbackWithSessionID() method e.g. depending on how the application provides its call reference.

Case a:

From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.

Case b:

The callEventNotify() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted.

In case the callEventNotify() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID(). See example in 4.6

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 setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

This parameter will be null if the notification is in NOTIFY mode and in case b).

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`

End of Change in Clause 6.2
End Of Document

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040720**
Meeting #29, Barcelona, SPAIN, 01-05 November 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-3**	**CR**	**032**	⌘**rev**	**-**	⌘	Current version:	**5.8.0**	⌘

*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:	⌘	Correct Behaviour of CallBack sequence and timing		
Source:	⌘	CN5 AePONA (Eamonn Murray)		
Work item code:	⌘	OSA1	***Date:*** ⌘	05/11/2004
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: ⌘ Misunderstandings in how to treat and use the callback functionality supported in call control services has been reported from the second OSA/Parlay PLUGTEST event. In particular the sequence and timing of specifying callbacks has been subject to different interpretations amongst vendors of applications and services. This was recognised as a major interoperability problem at the second OSA/Parlay Interoperability test.

As a result, CRs to improve the description of callback behaviour were introduced during CN5#27 as contributions N5-040338 through N5-040342. However the resulting specification text that has been produced, retains significant ambiguities regarding the use of the callback functionality, such that interoperability between application and service implementations, using the callback features, cannot be clearly understood.

It is therefore recommended to further clarify the description of the use of the callback features such that a clear and common understanding is possible for vendors of applications and services.

Summary of change: ⌘ Correctly define the use of callback features through text corrections to existing method semantics and correction to existing sequence diagram.

Consequences if not approved: ⌘ Interoperability cannot be supported, as the existing specification shall remain open to mutliple, differing, interpretations.

Clauses affected: ⌘ 6.1, 6.2

		Y	**N**			
Other specs affected:	⌘	X		Other core specifications	⌘	Rel-6 29.198-04-3
			X	Test specifications		
			X	O&M Specifications		

Other comments: ⌘ Rel-5 Mirror CR to N5-040718 Rel 4 CR 29.198-04

Change in Clause 6.1

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.

<<Interface>>

IpMultiPartyCallControlManager

createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier

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

destroyNotification (assignmentID : in TpAssignmentID) : void

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

<<deprecated>> getNotification () : TpNotificationRequestedSet

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

<<new>> enableNotifications (appCallControlManager : in IpAppMultiPartyCallControlManagerRef) : TpAssignmentID

<<new>> disableNotifications () : void

<<new>> getNextNotification (reset : in TpBoolean) : TpNotificationRequestedSetEntry

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~~ shall invoke setCallback() prior to createCall() 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, 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.

Setting the callback reference:

The call back reference can be registered either in a) createNotication() or b) explicitly with a setCallback() method e.g. depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the createNotification() with explicit registration may be the preferred method.

Case b:

The createNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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().

Set additional callback:

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 previously 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.

Setting the callback reference:

The callback reference can be registered either a) in enableNotications() or b) explicitly with a setCallback() method e.g. depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the createNotification() with explicit registation may be the preferred method.

Case b::

The enableNotifications() with no callback reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

In case the ~~createNotification~~enableNotifications() 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().

Set additional Call back:

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.

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 provisioned 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 network. Repeated 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 previously via the setCallback() method.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

End of Change in Clause 6.1

Change in Clause 6.2

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.

<<Interface>>

IpAppMultiPartyCallControlManager

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

callAborted (callReference : in TpSessionID) : void

managerInterrupted () : void

managerResumed () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

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.

Setting the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates. However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode.

When reportNotification() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. createAndRouteCallLegReq() is performed until the callback interface for the new call and/or new call leg has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation, or via the return of the reportNotification() method.

The call back reference can be registered either in a) reportNotification() or b) explicitly with a setCallbackWithSessionID() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the reportNotification() with explicit pass of registration may be the preferred method.

Case b:

The reportNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID(). If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted

In case reportNotification() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID().

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 setCallbackWithSessionID() invocation, this parameter may be set to P_APP_CALLBACK_UNDEFINED, or if supplied must be the same as that provided during the setCallbackWithSessionID().

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

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`

End of Change in Clause 6.2
End Of Document

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-2 CR 026** ⌘ **rev** **-** ⌘ Current version: **6.2.0** ⌘

*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: ⌘	Correct Behaviour of CallBack sequence and timing		
Source: ⌘	CN5 AePONA (Eamonn Murray)		
Work item code: ⌘	OSA1	***Date:*** ⌘	05/11/2004
Category: ⌘	**A**	***Release:*** ⌘	*REL-6*

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: ⌘ Misunderstandings in how to treat and use the callback functionality supported in call control services has been reported from the second OSA/Parlay PLUGTEST event. In particular the sequence and timing of specifying callbacks has been subject to different interpretations amongst vendors of applications and services. This was recognised as a major interoperability problem at the second OSA/Parlay Interoperability test.

As a result, CRs to improve the description of callback behaviour were introduced during CN5#27 as contributions N5-040338 through N5-040342. However the resulting specification text that has been produced, retains significant ambiguities regarding the use of the callback functionality, such that interoperability between application and service implementations, using the callback features, cannot be clearly understood.

It is therefore recommended to further clarify the description of the use of the callback features such that a clear and common understanding is possible for vendors of applications and services.

Summary of change: ⌘ Correctly define the use of callback features through text corrections to existing method semantics and correction to existing sequence diagram.

Consequences if not approved: ⌘ Interoperability cannot be supported, as the existing specification shall remain open to mutliple, differing, interpretations.

Clauses affected: ⌘ 4.6, 6.1, 6.2,

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	

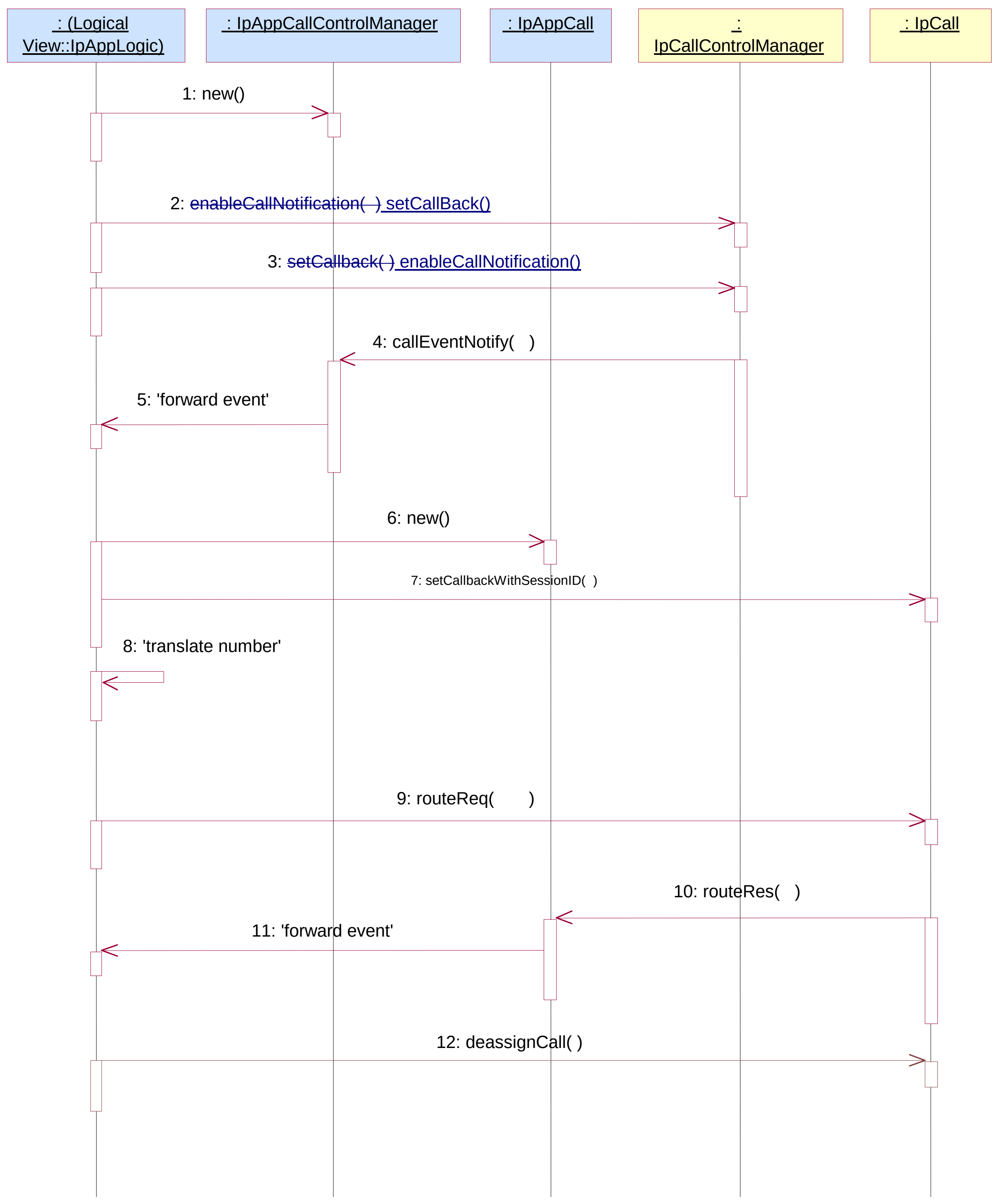
Other comments: ⌘ Rel-6 Mirror CR to N5-040718 Rel 4 CR 29.198-04

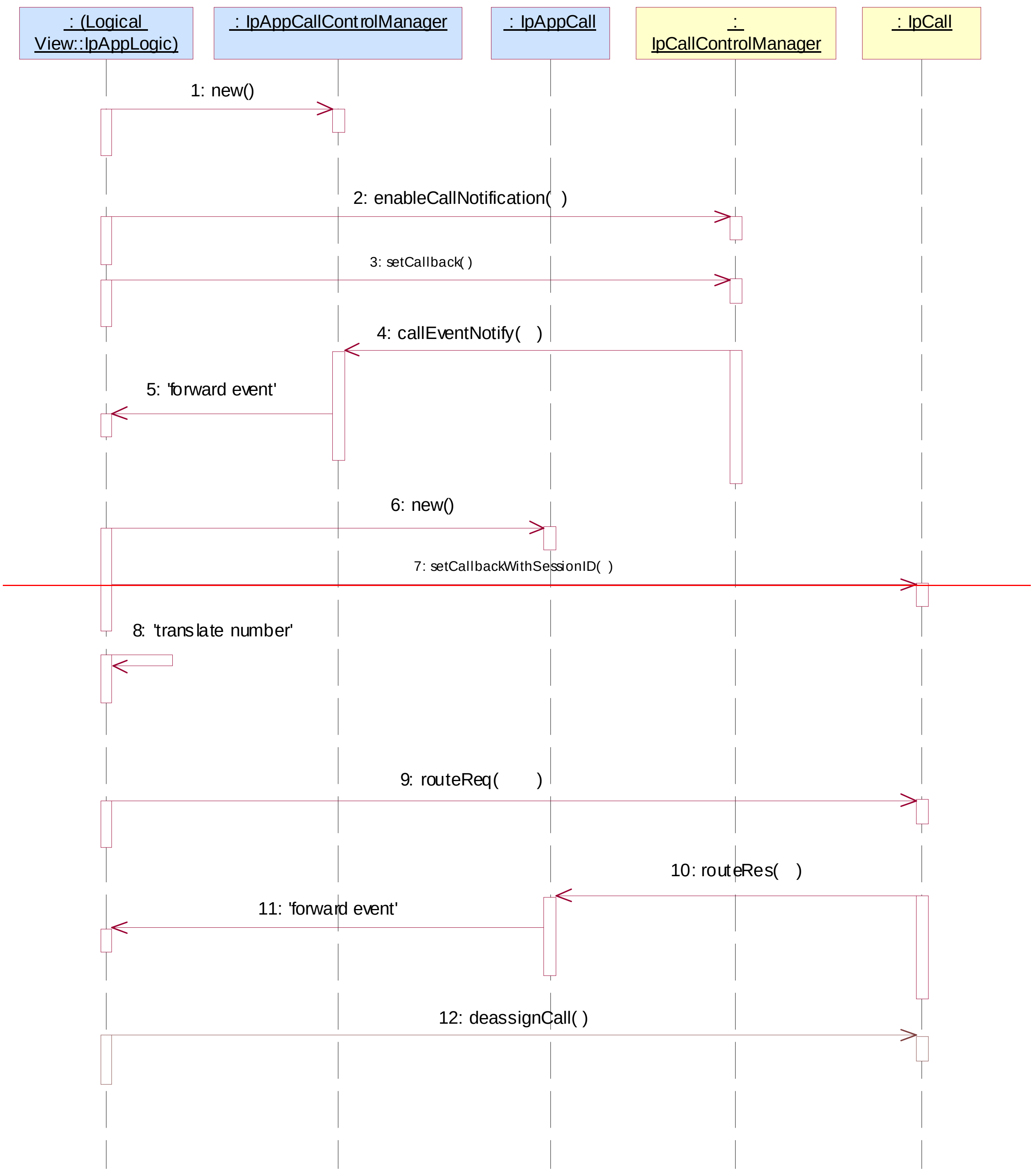
Change in Clause 4.6

4.6 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.

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.


: (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpCallControlManager
: IpCall
1: new()
2: ~~enableCallNotification( )~~ setCallBack()
3: ~~setCallback( )~~ enableCallNotification()
4: callEventNotify( )
5: 'forward event'
6: new()
7: setCallbackWithSessionID( )
8: 'translate number'
9: routeReq( )
10: routeRes( )
11: 'forward event'
12: deassignCall( )



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.~~ 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.

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.~~ 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 3, 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.

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.

End of Change in Clause 4.6

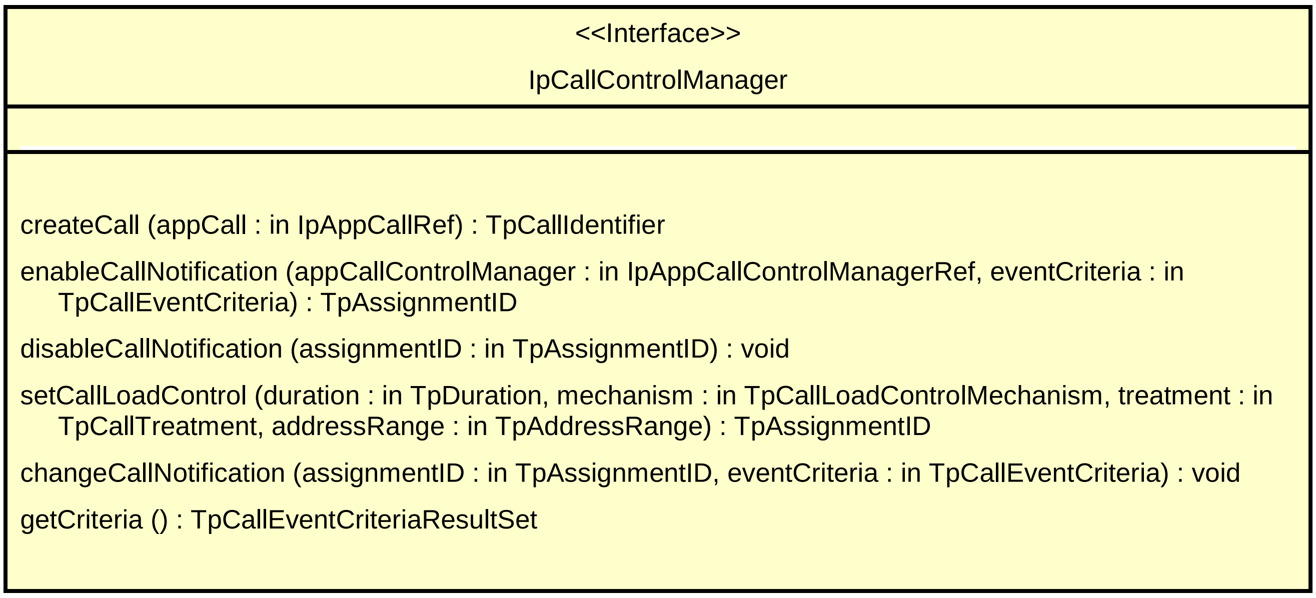
Change in Clause 6.1

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.



```
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.

Call back reference:

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~~ shall invoke setCallback() prior to createCall 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.

Setting the callback reference:

The call back reference can be registered either in a) enableCallNotification() or b) explicitly with a separate setCallback() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the enableCallNotification() with explicit immediate registration (no "Null" value) of call back reference may be the preferred method.

Case b:

The enableCallNotfication() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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(). See example in 6.1.6

Set additional callback:

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. See examples in 6.1.1.

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 previously 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`

End of Change in Clause 6.1

Change in Clause 6.2

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).

Setting the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates. However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode. When callEventNotify() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. routeReq() is performed until an IpAppCall has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

The call back reference can be registered either in a) callEventNotify() or b) explicitly with a setCallbackWithSessionID() method e.g. depending on how the application provides its call reference.

Case a:

From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.

Case b:

The callEventNotify() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted.

In case the callEventNotify() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID(). See example in 4.6

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 setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

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

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`

End of Change in Clause 6.2
End Of Document

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040722**
Meeting #29, Barcelona, SPAIN, 01-05 November 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3 CR 033** ⌘ **rev -** ⌘ Current version: **6.3.0** ⌘

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: ⌘	Correct Behaviour of CallBack sequence and timing
Source: ⌘	CN5 AePONA (Eamonn Murray)
Work item code: ⌘	OSA1
Date: ⌘	05/11/2004
Category: ⌘	**A**
Release: ⌘	*REL-6*

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: ⌘ Misunderstandings in how to treat and use the callback functionality supported in call control services has been reported from the second OSA/Parlay PLUGTEST event. In particular the sequence and timing of specifying callbacks has been subject to different interpretations amongst vendors of applications and services. This was recognised as a major interoperability problem at the second OSA/Parlay Interoperability test.

As a result, CRs to improve the description of callback behaviour were introduced during CN5#27 as contributions N5-040338 through N5-040342. However the resulting specification text that has been produced, retains significant ambiguities regarding the use of the callback functionality, such that interoperability between application and service implementations, using the callback features, cannot be clearly understood.

It is therefore recommended to further clarify the description of the use of the callback features such that a clear and common understanding is possible for vendors of applications and services.

Summary of change: ⌘ Correctly define the use of callback features through text corrections to existing method semantics and correction to existing sequence diagram.

Consequences if not approved: ⌘ Interoperability cannot be supported, as the existing specification shall remain open to mutliple, differing, interpretations.

Clauses affected: ⌘ 6.1, 6.2

Other specs affected: ⌘

Y	N		
	X	Other core specifications	⌘
	X	Test specifications	
	X	O&M Specifications	

Other comments: ⌘ Rel-6 Mirror CR to N5-040718 Rel 4 CR 29.198-04

Change in Clause 6.1

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.

<<Interface>> IpMultiPartyCallControlManager
createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier createNotification (appCallControlManager : in IpAppMultiPartyCallControlManagerRef, notificationRequest : in TpCallNotificationRequest) : TpAssignmentID destroyNotification (assignmentID : in TpAssignmentID) : void changeNotification (assignmentID : in TpAssignmentID, notificationRequest : in TpCallNotificationRequest) : void <<deprecated>> getNotification () : TpNotificationRequestedSet setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange) : TpAssignmentID enableNotifications (appCallControlManager : in IpAppMultiPartyCallControlManagerRef) : TpAssignmentID disableNotifications () : void getNextNotification (reset : in TpBoolean) : TpNotificationRequestedSetEntry

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~~ shall invoke setCallback() prior to createCall() 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, 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 a notification is requested by an application with an event type that is mutually exclusive compared to existing requested event types, then there is no need to check against the rest of the criteria for overlap. An example could be one application that trigger on "user busy" together with another application that trigger on "answer" - both requests should be allowed as only one can occur on the same call or session.

The overlap criteria have been defined to prevent multiple points of control, leading to possible interaction problems in networks that have no multi service support. Notice that dynamic aspects cannot be taken into account in the overlap criteria check. Therefore where dynamic event arming from an application causes a persistent control relationship it can prevent other applications to be invoked in the case single point of application control applies in the network.

However, the criteria check for overlap may as a network option be overruled by Multi Service networks allowing more services or applications to gain control of the same call or session at the same point in time. Refer to Call Control Common Definitions subpart of this specification (TS 29.198-4-1) for further details on application control over a call or session.

Setting the callback reference:

The call back reference can be registered either in a) createNotication() or b) explicitly with a setCallback() method e.g. depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the createNotification() with explicit registration may be the preferred method.

Case b:

The createNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

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().

Setting additional callback:

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 previously 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 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.

Setting the callback reference:

The callback reference can be registered either in a) enableNotications() or b) explicitly with a setCallback() method e.g. depending on how the application provides its callback reference.

Case a:

For an efficiency point of view the createNotification() with explicit registration may be the preferred method.

Case b:

The enableNotifications() with no callback reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided ~~subsequently~~ previously in a setCallback().If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised.

In case the ~~createNotification~~enableNotifications() 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().

Setting additional Call back:

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.

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 provisioned 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 previously via the setCallback() method.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

End of Change in Clause 6.1

Change in Clause 6.2

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.

<<Interface>>

IpAppMultiPartyCallControlManager

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

callAborted (callReference : in TpSessionID) : void

managerInterrupted () : void

managerResumed () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

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.

Setting the callback reference:

A reference to the application interface has to be passed back to the call interface to which the notification relates.

However, the setting of a call back reference is only applicable if the notification is in INTERRUPT mode.

When reportNotification() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. createAndRouteCallLegReq() is performed until the callback interface for the new call and/or new call leg has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation, or via the return of the reportNotification() method.

The call back reference can be registered either in a) reportNotification() or b) explicitly with a setCallbackWithSessionID() method depending on how the application provides its callback reference.

Case a:

From an efficiency point of view the reportNotification() with explicit pass of registration may be the preferred method.

Case b:

The reportNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided ~~subsequently~~ previously in a setCallbackWithSessionID(). If no callback reference has been provided previously to the service, the exception, P_NO_CALLBACK_ADDRESS_SET shall be raised, and no further application invocations related to the call shall be permitted

In case reportNotification() contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered previously by setCallbackWithSessionID().

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 setCallbackWithSessionID() invocation, this parameter may be set to P_APP_CALLBACK_UNDEFINED, or if supplied must be the same as that provided during the setCallbackWithSessionID().

This parameter will be set to P_APP_CALLBACK_UNDEFINED if the notification is in NOTIFY mode and in case b).

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`

End of Change in Clause 6.2
End Of Document