3GPP TSG-CN Meeting #24 **NP-040257**
02 – 04 June 2004, Seoul, KOREA

Source:	**CN5 (OSA)**
Title:	**5 Rel-4 CRs 29.198-xy OSA API: correct P_TRIGGERING_ADDRESSES service property**
Agenda item:	**7.10 (OSA Enhancements [OSA1])**
Document for:	**APPROVAL**

Doc-1st-	Spec	CR	Rev	Phase	Subject	Cat	Version	Doc-2nd-	Workite
NP-040257	29.198-04	069	-	Rel-4	Correction of callbacks sequence and timing conditions in GCCS and MPCCS	F	4.8.0	N5-040338	OSA1
NP-040257	29.198-04-2	016	-	Rel-5	Correction of callbacks sequence and timing conditions in GCCS	A	5.6.0	N5-040339	OSA1
NP-040257	29.198-04-2	017	-	Rel-6	Correction of callbacks sequence and timing conditions in GCCS	A	6.0.1	N5-040341	OSA1
NP-040257	29.198-04-3	025	-	Rel-5	Correction of callbacks sequence and timing conditions in MPCCS	A	5.6.0	N5-040340	OSA1
NP-040257	29.198-04-3	026	-	Rel-6	Correction of callbacks sequence and timing conditions in MPCCS	A	6.1.0	N5-040342	OSA1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040339**
Meeting #27, Miami, FL, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-2**	**CR**	**016**	⌘ **rev**	**-**	⌘	Current version:	**5.6.0** ⌘

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

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

Title:	⌘	Correction of callbacks sequence and timing conditions in GCCS
Source:	⌘	CN5 Parlay Appium
Work item code:	⌘	OSA1
Date:	⌘	14/05/2004
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 call backs has been reported from the second OSA/Parlay PLUGTEST event (N5-040077). The result of OSA/Parlay interoperability test reports major misunderstandings of how call back references were passed to Gateway for GCCS. Especially the sequence and timing conditions for sending call backs are subject for different interpretations among vendors. This has been recognised as a major problem at the second OSA/Parlay Interoperability test
Summary of change:	⌘	To solve the above problem, we therefore propose to introduce clarifying text for the sequence and timing of event for the sending of call backs for GCCS.
Consequences if not approved:	⌘	Interoperability problems

Clauses affected:	⌘	6.1, 6.2

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

Other comments:	⌘	This is a Rel-5 mirror to the CR in N5-040338

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. T~~(~~he application should 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.
Set of the callback reference:
The call back reference can be registered either in a) enableCallNotification() or b) explicit with a separate setCallback() method depending on how the application provides it's 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 in a setCallback().
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 4.6

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.
~~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 4.1

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 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).
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 the~~is~~ callEventNotify() method 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) explicit with a setCallbackWithSessionID() method e.g. depending on how the application provides it's call reference.
Case a:
From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.
The callEventNotify() method r~~R~~eturns 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 parameterwill be null if the notification is in NOTIFY mode and in case b).
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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID(). See example in 4.6

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

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	037	-	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	038	-	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	039	-	Addition of support for Emergency Telecommunications Service	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	040	-	Addition of support for Network Controlled Notifications MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	041	-	Changes to getNotification()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	042	-	Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	043	-	Addition of CAMEL Phase 4 Service Property values	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	044	-	Addition of indication whether SCS supports initially multiple routeReqs in parallel	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	045	-	Explicit exception for continueProcessing when not in interrupted mode	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	046	-	Indication needed that supervision will be ended when call or callLeg is deassigned	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	047	-	Clarify ambiguous Supervision duration	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	048	-	Detach/Attach request illegal during pending Attach/Detach request	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	049	-	Correction of Multi-Party Call Control properties	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	050	-	Correcting the sequence diagram descriptions in GCC and MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	051	-	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	052	-	Correcting the descriptions of sequence diagrams that don't match	4.4.0	5.0.0

					the diagram		
Jun 2002	CN_16	NP-020187	053	-	Correcting erroneous references to GCC in MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	054	-	Addition of the Multi-media APIs to Call control SCF (29.198-4)	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	055	-	Updating Clause 4 for Release 5	4.4.0	5.0.0
Jun 2002	CN_16	NP-020188	056	-	Spliting of 29.198-04 into 4 separate TSs (sub-parts)	4.4.0	5.0.0
Sep 2002	CN_17	NP-020430	001	--	29.198-04-2 Correction on use of NULL in Call Control API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	002	--	Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030020	003	-	Correction of status of GCC methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	004	-	Correction to Prepaid Sequence Diagram	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	005	-	Correction to TpCallEventCriteriaResult in Generic Call Control	5.1.0	5.2.0
Jun 2003	CN_20	NP-030238	007	--	Correction of the description for callEventNotify & reportNotification	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	008	--	Correction to Java Realisation Annex	5.3.0	5.4.0
Dec 2003	CN_22	NP-030544	009	--	Correction of description in superviseCallRes	5.4.0	5.5.0
Apr 2004	CN_23bis	NP-040155	011	--	Correct Java Code to conform with Java Rulebook in TS 29.198-01 and to remove errors	5.5.0	5.6.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040341**
Meeting #27, Miami, FL, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-2 CR 017** ⌘ **rev - ** ⌘ Current version: **6.0.1** ⌘

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

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

Title:	⌘	Correction of callbacks sequence and timing conditions in GCCS
Source:	⌘	CN5 Parlay Appium
Work item code:	⌘	OSA1
Date:	⌘	14/05/2004
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 call backs has been reported from the second OSA/Parlay PLUGTEST event (N5-040077). The result of OSA/Parlay interoperability test reports major misunderstandings of how call back references were passed to Gateway for GCCS. -Especially the sequence and timing conditions for sending call backs are subject for different interpretations among vendors. This has been recognised as a major problem at the second OSA/Parlay Interoperability test
Summary of change:	⌘	To solve the above problem, we therefore propose to introduce clarifying text for the sequence and timing of event for the sending of call backs for GCCS.
Consequences if not approved:	⌘	Interoperability problems

Clauses affected:	⌘	6.1, 6.2

Other specs affected: ⌘

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

Other comments:	⌘	This is a Rel-6 mirror to the CR in N5-040338

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.
Callback 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. ~~(~~T~~t~~he application should 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.
Set of the callback reference:
The call back reference can be registered either in a) enableCallNotification() or b) explicit with a separate setCallback() method depending on how the application provides it's 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 in a setCallback().
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 4.6

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 example in 4.1

~~In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().~~
Returns assignmentID: Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Parameters

`appCallControlManager : in IpAppCallControlManagerRef`

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

`eventCriteria : in TpCallEventCriteria`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE`

6.1.3 Method disableCallNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

Specifies the assignment ID given by the generic call control manager interface when the previous enableCallNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised. If two callbacks have been registered under this assignment ID both of them will be disabled.

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID`

6.1.4 Method setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the generic call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.
Returns assignmentID: Specifies the assignmentID assigned by the gateway to this request. This assignmentID can be used to correlate the callOverloadEncountered and callOverloadCeased methods with the request.

Parameters

`duration : in TpDuration`

Specifies the duration for which the load control should be set.
A duration of 0 indicates that the load control should be removed.
A duration of -1 indicates an infinite duration (i.e., until disabled by the application)
A duration of -2 indicates the network default duration.

`mechanism : in TpCallLoadControlMechanism`

Specifies the load control mechanism to use (for example, admit one call per interval), and any necessary parameters, such as the call admission rate. The contents of this parameter are ignored if the load control duration is set to zero.

`treatment : in TpCallTreatment`

Specifies the treatment of calls that are not admitted. The contents of this parameter are ignored if the load control duration is set to zero.

`addressRange : in TpAddressRange`

Specifies the address or address range to which the overload control should be applied or removed.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN`

6.1.5 Method changeCallNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the generic call control manager interface for the event notification. If two call backs have been registered under this assignment ID both of them will be changed.

`eventCriteria : in TpCallEventCriteria`

Specifies the new set of event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

6.1.6 Method getCriteria()

This method is used by the application to query the event criteria set with enableCallNotification or changeCallNotification.
Returns eventCriteria: Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

Parameters
No Parameters were identified for this method

Returns

`TpCallEventCriteriaResultSet`

Raises

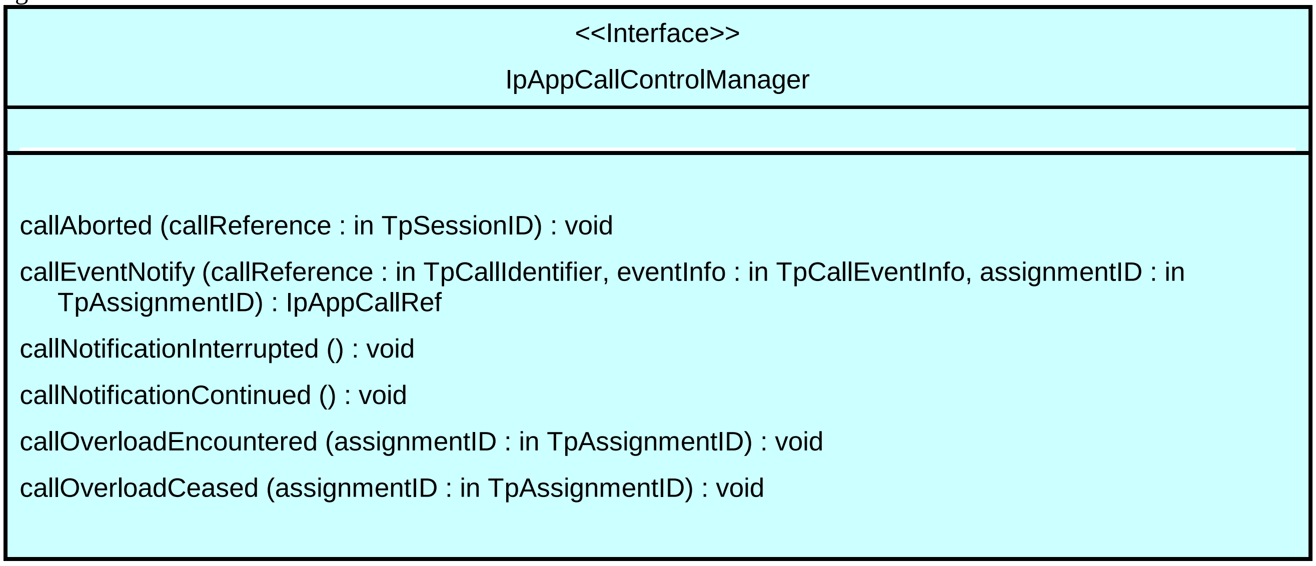
`TpCommonExceptions`

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.



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).
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 the~~is~~ callEventNotify() method 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) explicit with a setCallbackWithSessionID() method e.g. depending on how the application provides it's call reference.

Case a:
From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.
The callEventNotify() method r~~R~~eturns 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)..
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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID(). See example in 4.6

Parameters

callReference : in TpCallIdentifier

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

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

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

Returns

IpAppCallRef

6.2.3 Method callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporarily interrupted (for example, due to faults detected).
Note that more permanent failures are reported via the Framework (integrity management).

Parameters

No Parameters were identified for this method

6.2.4 Method callNotificationContinued()

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

Parameters

No Parameters were identified for this method

6.2.5 Method callOverloadEncountered()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.6 Method callOverloadCeased()

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

Parameters

`assignmentID : in TpAssignmentID`

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

End of Change in Clause 6.2

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	037	-	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	038	-	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	039	-	Addition of support for Emergency Telecommunications Service	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	040	-	Addition of support for Network Controlled Notifications MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	041	-	Changes to getNotification()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	042	-	Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	043	-	Addition of CAMEL Phase 4 Service Property values	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	044	-	Addition of indication whether SCS supports initially multiple routeReqs in parallel	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	045	-	Explicit exception for continueProcessing when not in interrupted mode	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	046	-	Indication needed that supervision will be ended when call or callLeg is deassigned	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	047	-	Clarify ambiguous Supervision duration	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	048	-	Detach/Attach request illegal during pending Attach/Detach request	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	049	-	Correction of Multi-Party Call Control properties	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	050	-	Correcting the sequence diagram descriptions in GCC and MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	051	-	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	052	-	Correcting the descriptions of sequence diagrams that don't match	4.4.0	5.0.0

					the diagram		
Jun 2002	CN_16	NP-020187	053	-	Correcting erroneous references to GCC in MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	054	-	Addition of the Multi-media APIs to Call control SCF (29.198-4)	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	055	-	Updating Clause 4 for Release 5	4.4.0	5.0.0
Jun 2002	CN_16	NP-020188	056	-	Spliting of 29.198-04 into 4 separate TSs (sub-parts)	4.4.0	5.0.0
Sep 2002	CN_17	NP-020430	001	--	29.198-04-2 Correction on use of NULL in Call Control API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	002	--	Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030020	003	-	Correction of status of GCC methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	004	-	Correction to Prepaid Sequence Diagram	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	005	-	Correction to TpCallEventCriteriaResult in Generic Call Control	5.1.0	5.2.0
Jun 2003	CN_20	NP-030238	007	--	Correction of the description for callEventNotify & reportNotification	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	008	--	Correction to Java Realisation Annex	5.3.0	5.4.0
Dec 2003	CN_22	NP-030544	009	--	Correction of description in superviseCallRes	5.4.0	5.5.0
Dec 2003	CN_22	NP-030553	010	--	Add OSA API support for 3GPP2 networks	5.5.0	6.0.0
Feb 2004	--	--	--	--	Added Java code attachment 2919804-2J2EE.zip which was delivered late by outside developers. See Annex C.	6.0.0	6.0.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040340**
Meeting #27, Miami, FL, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-3**	**CR**	**025**	⌘ **rev**	**-**	⌘	Current version:	**5.6.0** ⌘

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

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

Title: ⌘	Correction of callbacks sequence and timing conditions in MPCCS		
Source: ⌘	CN5 Parlay Appium		
Work item code: ⌘	OSA1	***Date:*** ⌘	14/05/2004
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 call backs has been reported from the second OSA/Parlay PLUGTEST event (N5-040077). The result of OSA/Parlay interoperability test reports major misunderstandings of how call back references were passed to Gateway for MPCCS. Especially the sequence and timing conditions for sending call backs are subject for different interpretations among vendors. This has been recognised as a major problem at the second OSA/Parlay Interoperability test
Summary of change: ⌘	To solve the above problem, we therefore propose to introduce clarifying text for the sequence and timing of event for the sending of call backs for MPCCS.
Consequences if not approved: ⌘	Interoperability problems

Clauses affected: ⌘	6.1, 6.2				
	Y	**N**			
Other specs ⌘	**X**		Other core specifications	⌘	Rel-4 29.198-04 Rel-6 29.198-04-3
affected:		**X**	Test specifications		
		X	O&M Specifications		
Other comments: ⌘	This is a Rel-5 mirror to the CR in N5-040338				

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, or the enableNotifications() and disableNotifications() 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 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 or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap when it leads to more than one application controlling the call or session at the same point in time during call or session processing.
If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not allow control on a call to be passed over. Only one application can place an interrupt request if the criteria overlaps.

Set of the callback reference:
The call back reference can be registered either in a) createNotication() or b) explicit with a setCallback() method e.g. depending on how the application provides it's 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 in a setcallback().
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 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.
Set of the callback reference:
The call back reference can be registered either in a) enableNotications() or b) explicit with a setCallback() method e.g. depending on how the application provides it's 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 call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided subsequently in a setCallback().
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 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 via the setCallback() method.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

6.1.8 Method <<new>> disableNotifications()

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

Parameters
No Parameters were identified for this method

Raises

`TpCommonExceptions`

6.1.9 Method <<new>> getNextNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification. Since a lot of data can potentially be returned (which might cause problem in the middleware), this method must be used in an iterative way. Each method invocation may return part of the total set of notifications if the set is too large to return it at once. The reset parameter permits the application to indicate whether an invocation to getNextNotification is requesting more notifications from the total set of notifications or is requesting that the total set of notifications shall be returned from the beginning.
Returns notificationRequestedSetEntry: The set of notifications and an indication whether all off the notifications have been obtained or if more notifications are available that have not yet been obtained by the application. If no notifications exist, an empty set is returned and the final indication shall be set to TRUE.
Note that the (maximum) number of items provided to the application is determined by the gateway.

Parameters

`reset : in TpBoolean`

TRUE: indicates that the application is intended to obtain the set of notifications starting at the beginning.
FALSE: indicates that the application requests the next set of notifications that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.
The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in TpNotificationRequestedSetEntry, for the next call to this method reset shall be set to TRUE. P_TASK_REFUSED may

be thrown if these conditions are not met.

Returns

`TpNotificationRequestedSetEntry`

Raises

`TpCommonExceptions`

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.
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.
The call back reference can be registered either in a) reportNotification() or b) explicit with a setCallbackWithSessionID() method depending on how the application provides it's callback reference.
Case a:
From an efficiency point of view the reportNotification() with explicit pass of registration may be the preferred method,
The reportNotification() method r~~R~~eturns 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)..
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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID().

Parameters

`callReference : in TpMultiPartyCallIdentifier`

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

`callLegReferenceSet : in TpCallLegIdentifierSet`

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.
However, if the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

`notificationInfo : in TpCallNotificationInfo`

Specifies data associated with this event (e.g. the originating or terminating leg which reports the notification).

`assignmentID : in TpAssignmentID`

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

Returns

`TpAppMultiPartyCallBack`

6.2.2 Method callAborted()

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

Parameters

`callReference : in TpSessionID`

Specifies the sessionID of call that has aborted or terminated abnormally.

6.2.3 Method managerInterrupted()

This method indicates to the application that event notifications and method invocations have been temporarily interrupted (for example, due to network resources unavailable).
Note that more permanent failures are reported via the Framework (integrity management).

Parameters
No Parameters were identified for this method

6.2.4 Method managerResumed()

This method indicates to the application that event notifications are possible and method invocations are enabled.

Parameters

No Parameters were identified for this method

6.2.5 Method callOverloadEncountered()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.6 Method callOverloadCeased()

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

Parameters

`assignmentID : in TpAssignmentID`

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

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	037	-	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	038	-	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	039	-	Addition of support for Emergency Telecommunications Service	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	040	-	Addition of support for Network Controlled Notifications MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	041	-	Changes to getNotification()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	042	-	Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	043	-	Addition of CAMEL Phase 4 Service Property values	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	044	-	Addition of indication whether SCS supports initially multiple routeReqs in parallel	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	045	-	Explicit exception for continueProcessing when not in interrupted mode	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	046	-	Indication needed that supervision will be ended when call or callLeg is deassigned	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	047	-	Clarify ambiguous Supervision duration	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	048	-	Detach/Attach request illegal during pending Attach/Detach request	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	049	-	Correction of Multi-Party Call Control properties	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	050	-	Correcting the sequence diagram descriptions in GCC and MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	051	-	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	052	-	Correcting the descriptions of sequence diagrams that don't match	4.4.0	5.0.0

					the diagram		
Jun 2002	CN_16	NP-020187	053	-	Correcting erroneous references to GCC in MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	054	-	Addition of the Multi-media APIs to Call control SCF (29.198-4)	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	055	-	Updating Clause 4 for Release 5	4.4.0	5.0.0
Jun 2002	CN_16	NP-020188	056	-	Spliting of 29.198-04 into 4 separate TSs (sub-parts)	4.4.0	5.0.0
Sep 2002	CN_17	NP-020431	001		29.198-04-3 Correction of error in Call Forward on Busy sequence diagram	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	002		Correct inconsistencies in IpCallLeg state transition diagrams	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	003		Clarification of the overlapping criteria definition and eventType mapping to IN TDPs	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	004		Add support for Carrier selection	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	005		Correction on use of NULL in Call Control API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	006		Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030031	007	--	Correction of status of MPCC methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030031	008	--	Inconsistent description of use of secondary callback	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	009	--	Correction to TpReleaseCauseSet in Multi Party Call Control IDL	5.1.0	5.2.0
Mar 2003	CN_19	NP-030130	010	--	Correction of definition of the P_MAX_CALLLEGS_PER_CALL	5.1.0	5.2.0
Jun 2003	CN_20	NP-030238	011	--	Correction of the description for callEventNotify & reportNotification	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	014	--	Correction to Java Realisation Annex	5.3.0	5.4.0
Dec 2003	CN_22	NP-030544	015	--	Correction of description in superviseRes	5.4.0	5.5.0
Dec 2003	CN_22	NP-030550	016	--	Correction of description of TpNotificationRequestedSetEntry	5.4.0	5.5.0
Apr 2004	CN_23bis	NP-040155	020	--	Correct Java Code to conform with Java Rulebook in TS 29.198-01 and to remove errors	5.5.0	5.6.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040342**
Meeting #27, Miami, FL, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3** **CR** **026** ⌘**rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title: ⌘	Correction of callbacks sequence and timing conditions in MPCCS		
Source: ⌘	CN5 Parlay Appium		
Work item code: ⌘	OSA1	***Date:*** ⌘	14/05/2004
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 call backs has been reported from the second OSA/Parlay PLUGTEST event (N5-040077). The result of OSA/Parlay interoperability test reports major misunderstandings of how call back references were passed to Gateway for MPCCS. Especially the sequence and timing conditions for sending call backs are subject for different interpretations among vendors. This has been recognised as a major problem at the second OSA/Parlay Interoperability test
Summary of change: ⌘	To solve the above problem, we therefore propose to introduce clarifying text for the sequence and timing of event for the sending of call backs for MPCCS.
Consequences if not approved: ⌘	Interoperability problems

Clauses affected: ⌘	6.1, 6.2			
	Y	**N**		
Other specs affected: ⌘		**X**	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	
Other comments: ⌘	This is a Rel-6 mirror to the CR in N5-040340			

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, or the enableNotifications() and disableNotifications() 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. T~~(~~he application should 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 or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap when it leads to more than one application controlling the call or session at the same point in time during call or session processing.
If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not allow control on a call to be passed over. Only one application can place an interrupt request if the criteria overlaps.
If 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.

Set of the callback reference:
The call back reference can be registered either in a) createNotication() or b) explicit with a setCallBack() method e.g. depending on how the application provides it's 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 in a setCallback().
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 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.

Set of the callback reference:
The call back reference can be registered either in a) enableNotications() or b) explicit with a setCallback() method e.g. depending on how the application provides it's 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 call back reference (“Null” value) is used where (e.g. due to distributed application logic) the call back reference is provided subsequently in a setCallback().
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 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 via the setCallback() method.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

6.1.8 Method <<new>> disableNotifications()

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

Parameters
No Parameters were identified for this method

Raises

`TpCommonExceptions`

6.1.9 Method <<new>> getNextNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification. Since a lot of data can potentially be returned (which might cause problem in the middleware), this method must be used in an iterative way. Each method invocation may return part of the total set of notifications if the set is too large to return it at once. The reset parameter permits the application to indicate whether an invocation to getNextNotification is requesting more notifications from the total set of notifications or is requesting that the total set of notifications shall be returned from the beginning.
Returns notificationRequestedSetEntry: The set of notifications and an indication whether all off the notifications have been obtained or if more notifications are available that have not yet been obtained by the application. If no notifications exist, an empty set is returned and the final indication shall be set to TRUE.
Note that the (maximum) number of items provided to the application is determined by the gateway.

Parameters

reset : in TpBoolean

TRUE: indicates that the application is intended to obtain the set of notifications starting at the beginning.
FALSE: indicates that the application requests the next set of notifications that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.
The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in TpNotificationRequestedSetEntry, for the next call to this method reset shall be set to TRUE. P_TASK_REFUSED may be thrown if these conditions are not met.

Returns

TpNotificationRequestedSetEntry

Raises

TpCommonExceptions

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.
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.
The call back reference can be registered either in a) reportNotification() or b) explicit with a setCallbackWithSessionID() method depending on how the application provides it's callback reference.
Case a:
From an efficiency point of view the reportNotification() with explicit pass of registration may be the preferred method,
The reportNotification() method r~~R~~eturns 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).
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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID().
.

Parameters

callReference : in TpMultiPartyCallIdentifier

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

callLegReferenceSet : in TpCallLegIdentifierSet

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.
However, if the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

notificationInfo : in TpCallNotificationInfo

Specifies data associated with this event (e.g. the originating or terminating leg which reports the notification).

assignmentID : in TpAssignmentID

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

Returns

TpAppMultiPartyCallBack

6.2.2 Method callAborted()

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

Parameters

callReference : in TpSessionID

Specifies the sessionID of call that has aborted or terminated abnormally.

6.2.3 Method managerInterrupted()

This method indicates to the application that event notifications and method invocations have been temporarily interrupted (for example, due to network resources unavailable).
Note that more permanent failures are reported via the Framework (integrity management).

Parameters

No Parameters were identified for this method

6.2.4 Method managerResumed()

This method indicates to the application that event notifications are possible and method invocations are enabled.

Parameters

No Parameters were identified for this method

6.2.5 Method callOverloadEncountered()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.6 Method callOverloadCeased()

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

Parameters

`assignmentID : in TpAssignmentID`

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

End of Change in Clause 6.2

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	037	-	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	038	-	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	039	-	Addition of support for Emergency Telecommunications Service	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	040	-	Addition of support for Network Controlled Notifications MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	041	-	Changes to getNotification()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	042	-	Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	043	-	Addition of CAMEL Phase 4 Service Property values	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	044	-	Addition of indication whether SCS supports initially multiple routeReqs in parallel	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	045	-	Explicit exception for continueProcessing when not in interrupted mode	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	046	-	Indication needed that supervision will be ended when call or callLeg is deassigned	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	047	-	Clarify ambiguous Supervision duration	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	048	-	Detach/Attach request illegal during pending Attach/Detach request	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	049	-	Correction of Multi-Party Call Control properties	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	050	-	Correcting the sequence diagram descriptions in GCC and MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	051	-	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	052	-	Correcting the descriptions of sequence diagrams that don't match	4.4.0	5.0.0

					the diagram		
Jun 2002	CN_16	NP-020187	053	-	Correcting erroneous references to GCC in MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	054	-	Addition of the Multi-media APIs to Call control SCF (29.198-4)	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	055	-	Updating Clause 4 for Release 5	4.4.0	5.0.0
Jun 2002	CN_16	NP-020188	056	-	Spliting of 29.198-04 into 4 separate TSs (sub-parts)	4.4.0	5.0.0
Sep 2002	CN_17	NP-020431	001		29.198-04-3 Correction of error in Call Forward on Busy sequence diagram	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	002		Correct inconsistencies in IpCallLeg state transition diagrams	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	003		Clarification of the overlapping criteria definition and eventType mapping to IN TDPs	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	004		Add support for Carrier selection	5.0.0	5.1.0
Sep 2002	CN_17	NP-020431	005		Correction on use of NULL in Call Control API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	006		Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030031	007	--	Correction of status of MPCC methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030031	008	--	Inconsistent description of use of secondary callback	5.1.0	5.2.0
Mar 2003	CN_19	NP-030020	009	--	Correction to TpReleaseCauseSet in Multi Party Call Control IDL	5.1.0	5.2.0
Mar 2003	CN_19	NP-030130	010	--	Correction of definition of the P_MAX_CALLLEGS_PER_CALL	5.1.0	5.2.0
Jun 2003	CN_20	NP-030238	011	--	Correction of the description for callEventNotify & reportNotification	5.2.0	5.3.0
Jun 2003	CN_20	NP-030305	012	1	Unclear overlap criteria for rejection of createNotification	5.3.0	6.0.0
Jun 2003	CN_20	NP-030247	013	--	Add support for advanced subscriber presentation	5.3.0	6.0.0
Dec 2003	CN_22	NP-030550	017	--	Correction of description of TpNotificationRequestedSetEntry	6.0.0	6.1.0
Dec 2003	CN_22	NP-030553	019	--	Add OSA API support for 3GPP2 networks	6.0.0	6.1.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040338**
Meeting #27, Miami, FL, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04 CR 069** ⌘ **rev** **-** ⌘ Current version: **4.8.0** ⌘

*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:	⌘	Correction of callbacks sequence and timing conditions in GCCS and MPCCS		
Source:	⌘	CN5 Parlay Appium		
Work item code:	⌘	OSA1	***Date:*** ⌘	14/05/2004
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 call backs has been reported from the second OSA/Parlay PLUGTEST event (N5-040077). The result of OSA/Parlay interoperability test reports major misunderstandings of how call back references were passed to Gateway for GCCS and MPCCS.
Especially the sequence and timing conditions for sending call backs are subject for different interpretations among vendors. This has been recognised as a major problem at the second OSA/Parlay Interoperability test

Summary of change: ⌘ To solve the above problem, we therefore propose to introduce clarifying text for the sequence and timing of event for the sending of call backs for GCCS as well as MPCCS.

Consequences if not approved: ⌘ Interoperability problems

Clauses affected: ⌘ 6.3.1, 6.3.2, 7.3.1 and 7.3.2

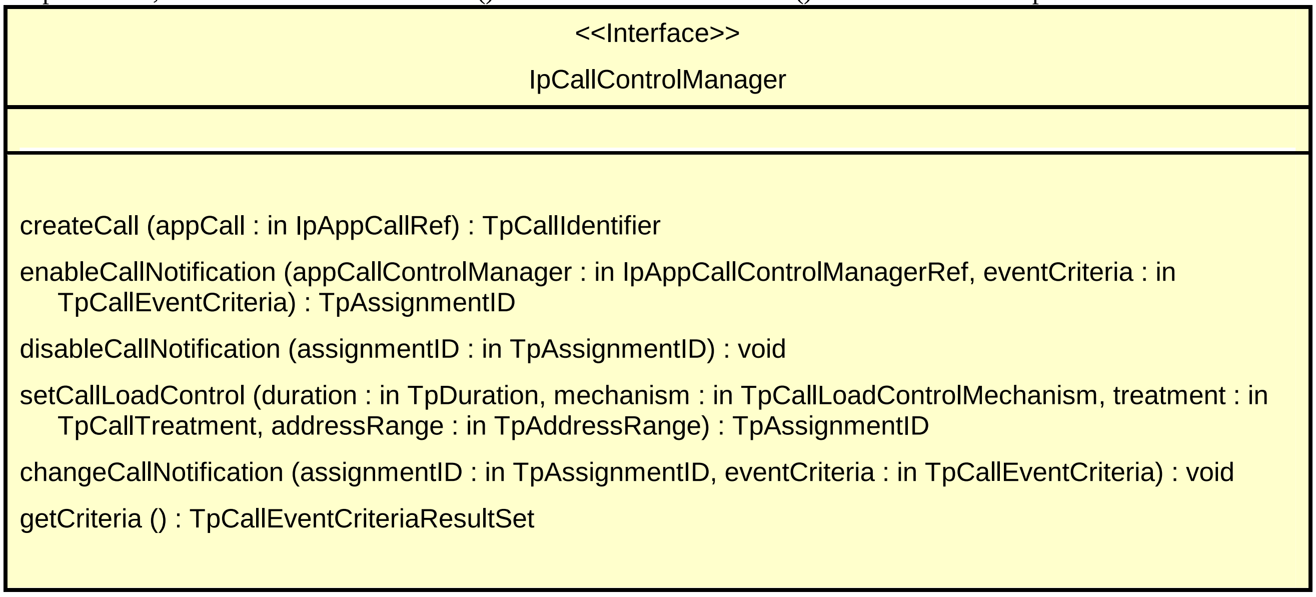
		Y	N			
Other specs affected:	⌘	X		Other core specifications	⌘	Rel-5/6 29.198-04-2 Rel-5/6 29.198-04-3
			X	Test specifications		
			X	O&M Specifications		

Other comments: ⌘ Mirror CRs to this CR exist for Rel-5 and Rel-6 in N5-040339 to N5-040342.

Change in Clause 6.3

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. T~~(t~~he application should 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) explicit with a separate setCallback() method depending on how the application provides it's 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 in a setCallback().
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
~~In case the enableCallNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().~~
Returns assignmentID: Specifies the ID assigned by the generic call control manager interface for this newly-enabled event notification.

Parameters

`appCallControlManager : in IpAppCallControlManagerRef`

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

`eventCriteria : in TpCallEventCriteria`

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

Returns

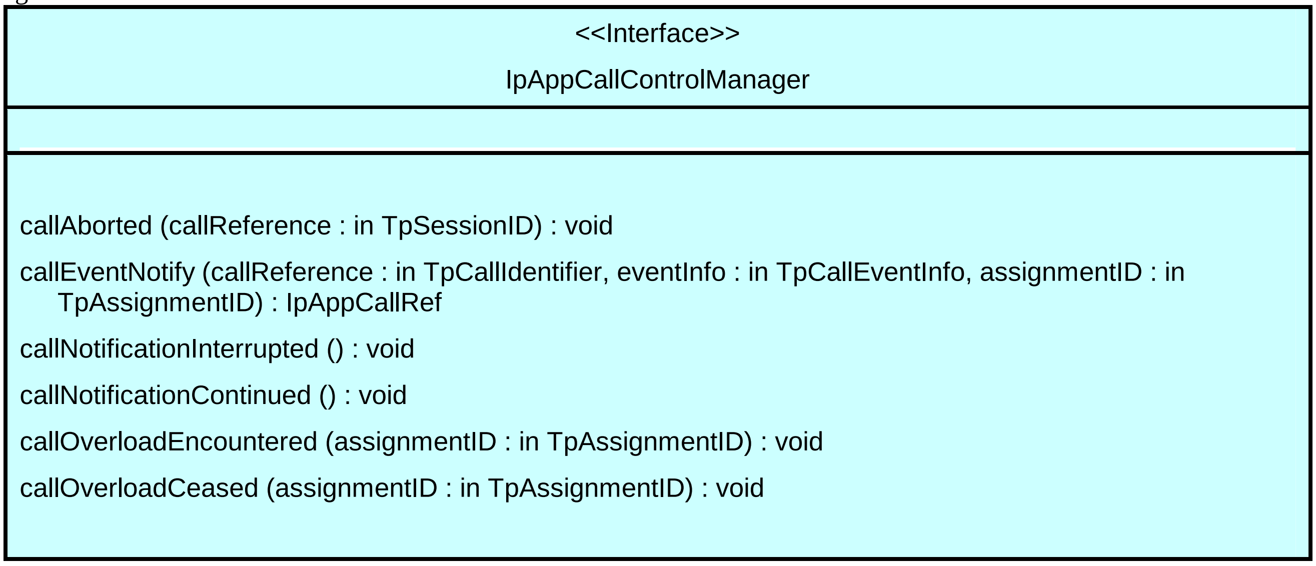
`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE`

6.3.2 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() ~~this method~~ 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) explicit with a setCallbackWithSessionID() method e.g. depending on how the application provides it's call reference.
Case a:
From an efficiency point of view the callEventNotify() with explicit pass of registration may be the preferred method.

The callEventNotify() methods ~~R~~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 .
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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID(). See example in 6.1.6

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

Change in Clause 7.3

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. ~~(~~T~~t~~he application should 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) explicit with a setCallback() method e.g. depending on how the application provides it's 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 in a setCallback().
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 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`

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.

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

Case a:

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

The reportNotification method() r~~R~~eturns 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)..

.

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 in a setCallbackWithSessionID().
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 by setCallbackWithSessionID().

Parameters

`callReference : in TpMultiPartyCallIdentifier`

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

`callLegReferenceSet : in TpCallLegIdentifierSet`

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.
However, if the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

`notificationInfo : in TpCallNotificationInfo`

Specifies data associated with this event (e.g. the originating or terminating leg which reports the notification).

`assignmentID : in TpAssignmentID`

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

Returns

`TpAppMultiPartyCallBack`

Method

`callAborted()`

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

Parameters

`callReference : in TpSessionID`

Specifies the sessionID of call that has aborted or terminated abnormally.

End of Change in Clause 7.3

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parmeter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Sep 2002	CN_17	NP-020424	057	--	Correction on use of NULL in Call Control API	4.4.0	4.5.0
Mar 2003	CN_19	NP-030020	058	--	Correction of status of methods to interfaces in clause 6.3	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	059	--	Correction to TpReleaseCauseSet in Multi Party Call Control	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	060	--	Correction to Sequence Diagrams to remove incorrect Framework references	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	061	--	Correction to User Interaction Prepaid Sequence Diagrams	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	062	--	Correction to remove unused TpCallChargeOrder	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	063	--	Correction to TpCallEventCriteriaResult in Generic Call Control	4.5.0	4.6.0
Mar 2003	CN_19	NP-030020	064	--	Correction of status of methods to interfaces in clause 7.3	4.5.0	4.6.0
Jun 2003	CN_20	NP-030238	065	--	Correction of the description for callEventNotify & reportNotification	4.6.0	4.7.0
Dec 2003	CN_22	NP-030544	066	--	Correction of description in superviseRes and superviseCallRes	4.7.0	4.8.0