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

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

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

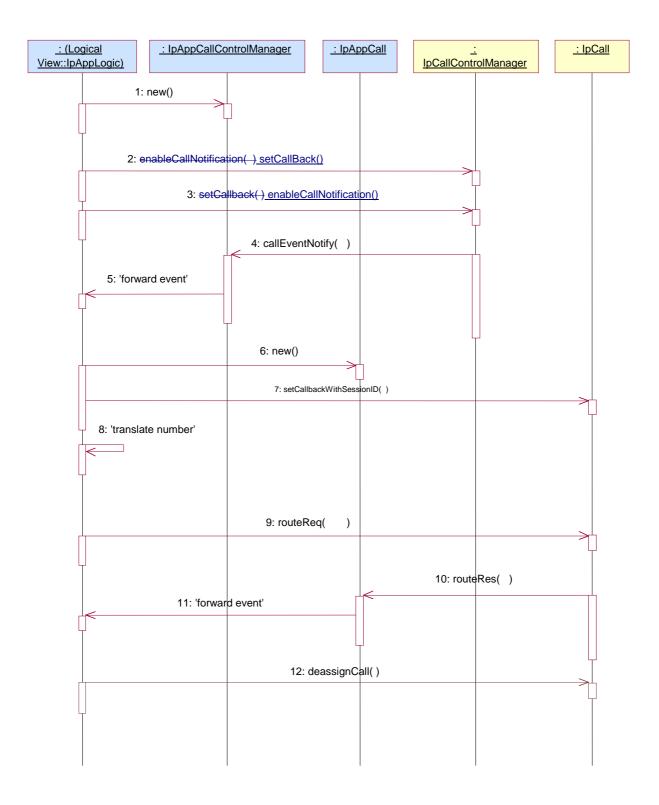
joint-API-group (Meeting #29, Ba	•								N W	G5)		I	N5-0407	18
				HAN									CR-Form	1-v7
[#] 29	<mark>.198</mark>	<mark>8-04</mark>	CR	071	ж	rev	-	ж	Curre	ent vers	sion:	<mark>4.10</mark>	^ж 0.	
For <u>HELP</u> on us	sing t	his for	m, see	bottom or	f this pa	age or le	ook	at the	e pop-	up text	over	the X	symbols.]
Proposed change a	affect	t s: L	JICC a	ops೫		ME	Rac	dio Ac	ccess	Netwo	rk	Core	Network	X
Title: ¥	Cor	rect Be	ehavio	ur of CallE	Back se	quence	anc	<mark>d timi</mark>	ng					
Source: ೫	CN	5 AePO	<mark>ona (</mark> e	amonn N	lurray)									
Work item code: ℜ	OS.	A1							D	ate: ೫	05/	<mark>11/200</mark>)4	
Category: ¥	Detai be fo	F (corr A (corr B (add C (func D (editi iled exp und in 3 Misur call c event subje This OSA/ As a introc Howe signif interc callba	ection) respond ition of ctional r orial mo lanation 3GPP <u>1</u> andersta ontrol s t. In pa ect to d was re- /Parlay result, duced of ever the icant a operabia	services h rticular the ifferent int cognised Interoper CRs to im during CN e resulting	how to bove cat how to bas bee e seque terpreta as a ma cability to 5#27 a g specif s regard en app hot be	tegories tegories tegories treat a n repor ence ar ajor inte test. the des s contri ication ding the lication clearly	can nd u ted 1 d tir mon cript butic text and und	ise th from g gst v erabil ion o ons N that I serv ersto	Use 2 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	2 R96 R97 R98 R99 Rel-4 Rel-5 Rel-6 back fu back fu back be D338 th Dack be D338 th Dack fu Dack	the fo (GSM (Rele (Rele (Rele (Rele (Rele (Rele (Rele (Rele Callba DSA/F callba plication callba plication callba plication callba call	llowing I Phase ase 19 ase 19 ase 19 ase 19 ase 5) ase 6) Parlay I acks h ions ar secon ur wer h N5-0- d, retai nality, ns, usi	96) 97) 98) 99) 99) PLUGTES as been nd service d e 40342. ins such that ng the	ST S.
				tures suc applicatior				comn	non ur	ndersta	nding	is pos	sible for	
Summary of chang	е: Ж			fine the un nantics an						-			to existing	
Consequences if not approved:	ж			ility canno liple, diffe					existin	g spec	ificatio	on sha	ll remain	
Clauses affected:	ж	6.1.6	<mark>, 6.3.1</mark> ,	6.3.2, 7.3	3.1, 7.3	.2								
Other specs affected:	Ħ		Test s	core spec pecification Specificat	ons	ns	ж	Rel-	5/6: 29	9.198-0)4-2, 2	29.198	-04-3	
Other comments:	ж			CRs in N CRs in N										

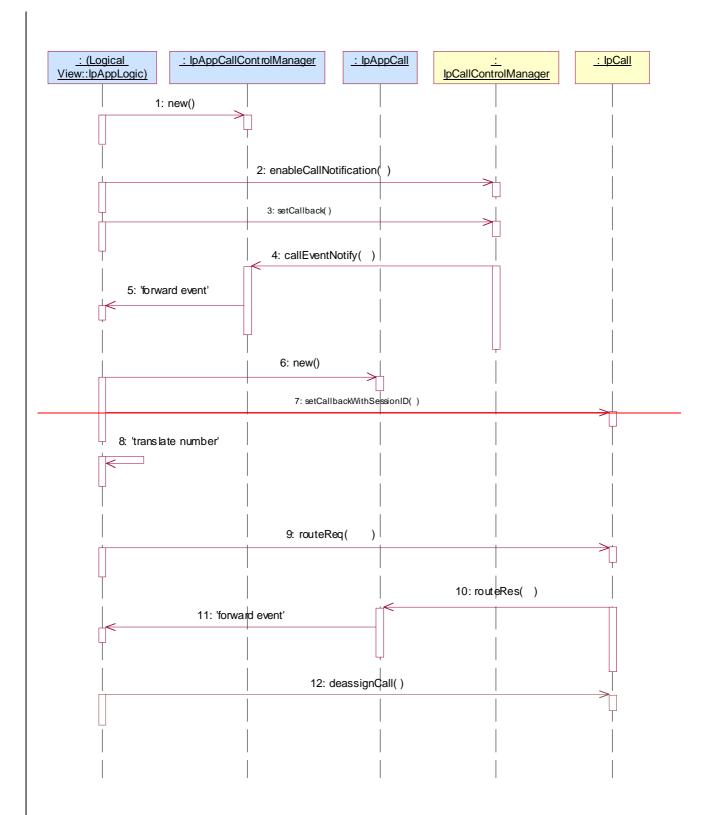
Change in Clause 6.1.6

6.1.6 Number Translation 1 (with callbacks)

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

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





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

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

enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification.

3: This message sets the reference of the IpAppCallControlManager object in the CallControlManager. The CallControlManager reports the callEventNotify to referenced object only for enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification. This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria set in message 3, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

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

- 5: This message is used to forward message 4 to the IpAppLogic.
- 6: This message is used by the application to create an object implementing the IpAppCall interface.
- 7: This message is used to set the reference to the IpAppCall for this call.
- 8: This message invokes the number translation function.

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

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

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

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

End of Change in Clause 6.1.6

Change in Clause 6.3.1

6.3.1 Interface Class IpCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Generic Call Control Service. The generic call control manager interface provides the management functions to the generic call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications.

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

<<Interface>>

IpCallControlManager

createCall (appCall : in IpAppCallRef) : TpCallIdentifier

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

disableCallNotification (assignmentID : in TpAssignmentID) : void

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

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

getCriteria () : TpCallEventCriteriaResultSet

Method createCall()

This method is used to create a new call object.

Call back reference:

An IpAppCallControlManager should already have been passed to the IpCallControlManager, otherwise the call control will not be able to report a callAborted() to the application. The application <u>should_shall</u> invoke setCallback() prior to createCall if it wishes to ensure this.

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

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

Method enableCallNotification()

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

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

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

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

Set of the callback reference:

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

Case a:

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

Case b:

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

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

Set additional callback reference:

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. See examples in 6.1.1.

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

Parameters

appCallControlManager : in IpAppCallControlManagerRef

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

eventCriteria : in TpCallEventCriteria

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

End of Change in Clause 6.3.1

Change in Clause 6.3.2

6.3.2 Interface Class IpAppCallControlManager

Inherits from: IpInterface

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

<<Interface>>

IpAppCallControlManager

callAborted (callReference : in TpSessionID) : void

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

callNotificationInterrupted () : void

callNotificationContinued () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

Method callAborted()

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

Parameters

callReference : in TpSessionID

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

Method callEventNotify()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

Set of the callback reference:

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

When callEventNotify() is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, the application writer should ensure that no continue processing e.g. routeReq() is performed until an IpAppCall has been

passed to the gateway, either through an explicit setCallbackWithSessionID() invocation on the supplied IpCall, or via the return of the callEventNotify() method.

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

Case a:

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

Case b:

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

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 <u>previously</u> by setCallbackWithSessionID(). See example in 6.1.6

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

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

Parameters

callReference : in TpCallIdentifier

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

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

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

Returns

IpAppCallRef

End of Change in Clause 6.3.2

Change in Clause 7.3.1

7.3.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Multi-party Call Control Service. The multi-party call control manager interface provides the management functions to the multi-party call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications. The action table associated with the STD shows in what state the IpMultiPartyCallControlManager must be if a method can successfully complete. In other words, if the IpMultiPartyCallControlManager is in another state the method will throw an exception immediately.

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

<<Interface>>

IpMultiPartyCallControlManager

createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier

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

destroyNotification (assignmentID : in TpAssignmentID) : void

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

getNotification () : TpNotificationRequestedSet

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

Method createCall()

This method is used to create a new call object. An IpAppMultiPartyCallControlManager should already have been passed to the IpMultiPartyCallControlManager, otherwise the call control will not be able to report a callAborted() to the application. The application should should should

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

Parameters

appCall : in IpAppMultiPartyCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpMultiPartyCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

Method createNotification()

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of calls happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the

context of a particular call session it has to use the createAndRouteCallLegReq() method on the call object or the eventReportReq() method on the call leg object. The application will get access to the call object when it receives the reportNotification(). (Note that createNotification() is not applicable if the call is setup by the application).

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

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

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

Set of the callback reference:

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

Case a:

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

Case b:

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

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

Set additional Call back:

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

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

Parameters

appCallControlManager : in IpAppMultiPartyCallControlManagerRef

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

notificationRequest : in TpCallNotificationRequest

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

End of Change in Clause 7.3.1

Change in Clause 7.3.2

7.3.2 Interface Class IpAppMultiPartyCallControlManager

Inherits from: IpInterface

The Multi-Party call control manager application interface provides the application call control management functions to the Multi-Party call control service.

< <interface>></interface>	
IpAppMultiPartyCallControlManager	

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

callAborted (callReference : in TpSessionID) : void

managerInterrupted () : void

managerResumed () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

Method reportNotification()

This method notifies the application of the arrival of a call-related event.

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

Set of the callback reference:

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

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

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

Case a:

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

Case b:

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

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

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

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

Parameters

callReference : in TpMultiPartyCallIdentifier

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

callLegReferenceSet : in TpCallLegIdentifierSet

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.

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

notificationInfo : in TpCallNotificationInfo

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

assignmentID : in TpAssignmentID

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

Returns

TpAppMultiPartyCallBack

End of Change in Clause 7.3.2

End Of Document

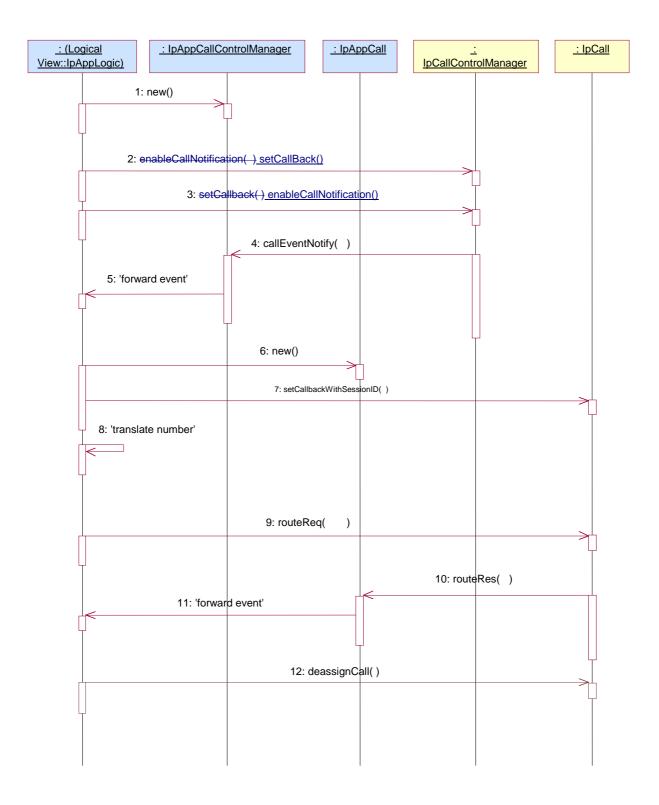
Meeting #	23, Da		ona, i							_				CR-Form-v7
				C	HAN	GE	REC	QUE	EST	-				
ж	29.1	<mark>98-</mark>	<mark>04-2</mark>	CR	025	:	⊭ rev	-	ж	Cur	rent ver	sion:	5.8.0	ж
For <mark>HEL</mark>	. <u>P</u> on u	sing t	this for	m, see	bottom o	of this	oage o	r look	at th	ne pop	o-up tex	t over	r the ℜ sy	mbols.
Proposed c	hange a	affec	<i>ts:</i> เ	JICC a	ops#]	ME	Ra	dio A	\cces	s Netwo	ork	Core N	etwork X
Title:	ж	Co	rect B	<mark>ehavio</mark>	ur of Cal	Back	sequen	<mark>ce an</mark>	<mark>id tim</mark>	ing				
Source:	ж	CN	5 AeP	<mark>ona (</mark> i	amonn	Murray	')							
Work item o	:ode: %	OS	A1								Date: ३	€ <mark>05</mark> ,	/11/2004	
Category:	ж	Α								Rel	ease: a	€ RE	L-5	
					wing cate	gories:					se <u>one</u> o	f the fo	ollowing re	
l				rection) respond	ls to a coi	rection	in an e	arlier r	eleas	e)	2 R96		M Phase 2 ease 1996	
			B (add	lition of	feature),					- /	R97	(Rele	ease 1997)
					nodificatio		ature)				R98		ease 1998	
					o <i>dification</i> ns of the a		ategori	es can	h		R99 Rel-4		ease 1999 _, ease 4))
					R 21.900		alogon	50 001	•		Rel-5		ease 5)	
											Rel-6	(Rele	ease 6)	
			even subje This OSA As a introd Howe signif intere callb	t. In pa ect to d was re /Parlay result, duced ever th ficant a operab ack fea herefo ack fea	rticular ti ifferent ir cognised Interope CRs to i during CI e resultir mbiguitie lity betw tures, ca e recom	he seq hterpred as a perability mprove N5#27 ng speces rega een ap annot b mende ch that	uence tations major in / test. e the d as cor cification e clean ed to fun a clean	and ti amounterop escription tribut on tex be us on and ly und rther r and	otion of ions l t that d sen clarify	of sp vendo vility p of cal N5-04 has l the ca vice in ood. y the	becifying broblem Iback b 40338 ti been pr allback mpleme	g callb oplicat at the ehavio hrough oduce functio entatio	acks has	342. ch that the of the
Summary o	f chang	յe: ೫			fine the nantics a								ections to n.	existing
Consequen not approve		Ħ			ility cann liple, diff					exist	ing spe	cificat	ion shall r	emain
Clauses aff	ected:	ж	46 (6.1, 6.2										
Siddood all		00	1.0, 0	, 0.2	,									
Other spec: affected:	S	ж	Y N X X X X X X	Test s	core spe pecificat Specifica	ions	ions	Ж	Rel	-6: 29	0.198-04	1-2		
Other comn	nents:	ж	Rel-	5 Mirro	CR to N	15-040	718 Re		R 20	198-0)4			

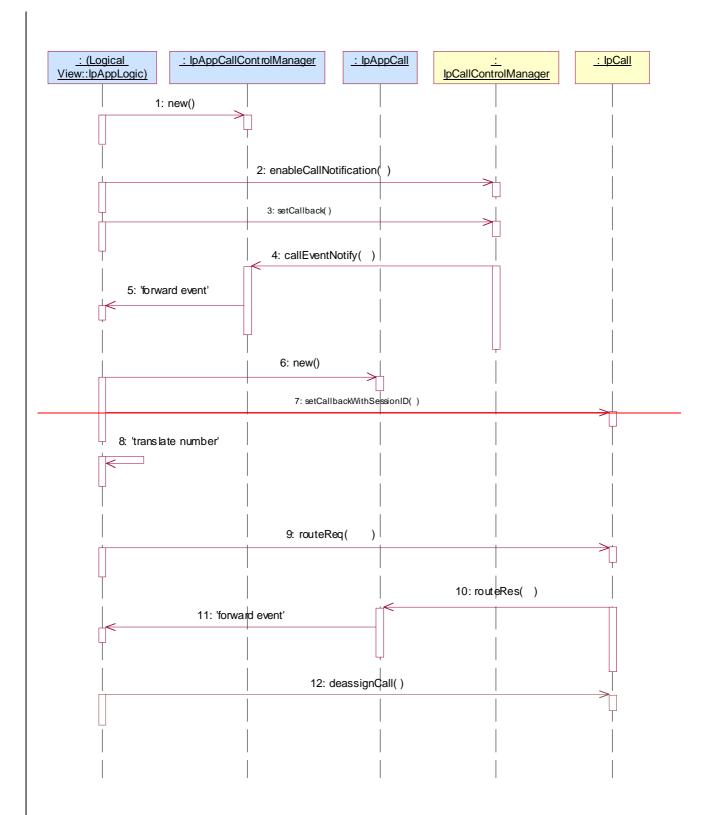
Change in Clause 4.6

4.6 Number Translation 1 (with callbacks)

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

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





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

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

enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification.

3: This message sets the reference of the IpAppCallControlManager object in the CallControlManager. The CallControlManager reports the callEventNotify to referenced object only for enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification. This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria set in message 3, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

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

- 5: This message is used to forward message 4 to the IpAppLogic.
- 6: This message is used by the application to create an object implementing the IpAppCall interface.
- 7: This message is used to set the reference to the IpAppCall for this call.
- 8: This message invokes the number translation function.

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

- 10: This message passes the result of the call being answered to its callback object
- 11: This message is used to forward the previous message to the IpAppLogic.

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

End of Change in Clause 4.6

Change in Clause 6.1

6.1 Interface Class IpCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Generic Call Control Service. The generic call control manager interface provides the management functions to the generic call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications.

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

<<Interface>>

IpCallControlManager

createCall (appCall : in IpAppCallRef) : TpCallIdentifier

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

disableCallNotification (assignmentID : in TpAssignmentID) : void

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

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

getCriteria () : TpCallEventCriteriaResultSet

6.1.1 Method createCall()

This method is used to create a new call object.

Call back reference:

An IpAppCallControlManager should already have been passed to the IpCallControlManager, otherwise the call control will not be able to report a callAborted() to the application. The application <u>should_shall</u> invoke setCallback() prior to createCall if it wishes to ensure this.

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

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 Method enableCallNotification()

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

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

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

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

Setting the callback reference:

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

Case a:

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

Case b:

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

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

Set additional callback:

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. See examples in 6.1.1.

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

Parameters

appCallControlManager : in IpAppCallControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified <u>previously</u> 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>></interface>
IpAppCallControlManager
callAborted (callReference : in TpSessionID) : void
callEventNotify (callReference : in TpCallIdentifier, eventInfo : in TpCallEventInfo, assignmentID : in TpAssignmentID) : IpAppCallRef
callNotificationInterrupted () : void
callNotificationContinued () : void
callOverloadEncountered (assignmentID : in TpAssignmentID) : void
callOverloadCeased (assignmentID : in TpAssignmentID) : void

6.2.1 Method callAborted()

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

Parameters

callReference : in TpSessionID

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

6.2.2 Method callEventNotify()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

Setting the callback reference:

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

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

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

Case a:

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

Case b:

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

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 <u>previously</u> by setCallbackWithSessionID(). See example in 4.6

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

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

Parameters

callReference : in TpCallIdentifier

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

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

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

Returns

IpAppCallRef

End of Change in Clause 6.2 End Of Document

joint-API-q Meeting #		•			•		•							-040720
				C	HAN	IGE	REC	QUE	ST					CR-Form-v7
æ	29.1	<mark>98-0</mark>	<mark>4-3</mark>	CR	032		≋rev	-	ж	Current v	ersio	n:	5.8.0	ж
For <mark>HEL</mark>	.P on us	sing th	nis for	m, see	bottom	of this	page o	r look	at th	e pop-up te	ext ov	/er t	he 🛱 syı	nbols.
Proposed c	hange a	affect	s: l	JICC a	ops₩		ME	Ra	dio A	ccess Net	work		Core Ne	etwork X
Title:	ж	Corr	ect B	ehavio	ur of Cal	Back s	sequen	<mark>ce an</mark>	d tim	ing				
Source:	Ħ	CN5	AeP	<mark>ona (</mark> e	amonn	Murray	/)							
Work item o	code:	OS/	\1							Date:	· ೫ (05/1	1/2004	
Category:	Ħ	Α								Release:	· ж	REL	-5	
		Use <u>o</u> F E C Detail	(corr (corr (add (fund (fund (edit ed exp	rection) respond lition of ctional r torial mo blanation	wing cate s to a confeature), modification dification ns of the R 21.900	on of fean on of fean on on of fean on)	in an ea ature)			Use <u>one</u> 2	of the (C (R (R (R (R (R (R (R	SSM Relea Relea Relea Relea Relea	owing rele Phase 2) ase 1996) ase 1997) ase 1998) ase 1999) ase 4) ase 5) ase 6)	
			even subje This OSA As a introd Howo signif intero callba It is t	t. In pa ect to d was re- /Parlay result, duced o ever the ficant a operabi ack fea herefor ack fea	rticular t fferent in cognised Interope CRs to i during C e resultir mbiguiti lity betw tures, ca e recom tures su	he seq nterpred d as a r erability mprove N5#27 ng spec es rega veen ap annot b mende ich that	uence tations major ir y test. e the de as con cificatio arding t oplicatio be clear ed to fu t a clea	and ti amor hterop tributi n text he us on and ly und rther o r and	ming ogst v oerab otion of ions I t that is of t d serv dersto	the secon of specifyi vendors of ility problem of callback N5-040338 has been he callbac vice implem ood.	ng ca applie m at t beha throu produ k fund nenta	allba catio he s viou ugh iced ction ition	acks has bons and s second N5-0403 I, retains hality, suc s, using the use o	been services. 342. ch that the of the
Summary o	f chang	е: Ж	Corre	ectly de		use of	callbad	<mark>ck fea</mark>		through te sequence				existing
Consequen not approve		ж			lity canr liple, dif					existing sp	ecific	atio	n shall re	emain
Clauses aff	ected:	ж	6.1, 6	6.2										
Other spec: affected:	S	ж	YN X X X X	Test s	core spo pecifica Specifica	tions	ions	ж	Rel-	<mark>6 29.198-0</mark>)4-3			
Other comm	nents:	ж	Rel-5	5 Mirror	CR to N	15-040	718 Re	I 4 CF	R 29.1	198-04				

Change in Clause 6.1

6.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Multi-party Call Control Service. The multi-party call control manager interface provides the management functions to the multi-party call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications. The action table associated with the STD shows in what state the IpMultiPartyCallControlManager must be if a method can successfully complete. In other words, if the IpMultiPartyCallControlManager is in another state the method will throw an exception immediately.

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

< <interface>></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</deprecated>
setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange) : TpAssignmentID
< <new>> enableNotifications (appCallControlManager : in IpAppMultiPartyCallControlManagerRef) : TpAssignmentID</new>
< <new>> disableNotifications () : void</new>
< <new>> getNextNotification (reset : in TpBoolean) : TpNotificationRequestedSetEntry</new>

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 should should

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

Parameters

appCall : in IpAppMultiPartyCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpMultiPartyCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 Method createNotification()

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

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

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

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

Setting the callback reference:

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

Case a:

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

Case b:

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

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

Set additional callback:

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

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

Parameters

appCallControlManager : in IpAppMultiPartyCallControlManagerRef

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

notificationRequest : in TpCallNotificationRequest

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

6.1.3 Method destroyNotification()

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

Parameters

assignmentID : in TpAssignmentID

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

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

6.1.4 Method changeNotification()

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

Parameters

assignmentID : in TpAssignmentID

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

notificationRequest : in TpCallNotificationRequest

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

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

6.1.5 Method <<deprecated>> getNotification()

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

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns notificationsRequested: Specifies the notifications that have been requested by the application. An empty set is returned when no notifications exist.

Parameters

No Parameters were identified for this method

Returns

TpNotificationRequestedSet

Raises

TpCommonExceptions

6.1.6 Method setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

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

Parameters

duration : in TpDuration

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

mechanism : in TpCallLoadControlMechanism

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

treatment : in TpCallTreatment

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

addressRange : in TpAddressRange

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

6.1.7 Method <<new>> enableNotifications()

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

Setting the callback reference:

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

Case a:

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

Case b::

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

In case the <u>createNotificationenableNotifications()</u> 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 <u>previously</u> via the setCallback() method.

Returns

TpAssignmentID

Raises

TpCommonExceptions

End of Change in Clause 6.1

Change in Clause 6.2

6.2 Interface Class IpAppMultiPartyCallControlManager

Inherits from: IpInterface

The Multi-Party call control manager application interface provides the application call control management functions to the Multi-Party call control service.

<<Interface>>

IpAppMultiPartyCallControlManager

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

callAborted (callReference : in TpSessionID) : void

managerInterrupted () : void

managerResumed () : void

callOverloadEncountered (assignmentID : in TpAssignmentID) : void

callOverloadCeased (assignmentID : in TpAssignmentID) : void

6.2.1 Method reportNotification()

This method notifies the application of the arrival of a call-related event.

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

Setting the callback reference:

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

When reportNotification() is invoked with a monitor mode of P CALL MONITOR MODE INTERRUPT, the application writer should ensure that no continue processing e.g. createAndRouteCallLegReq() is performed until the

callback interface for the new call and/or new call leg has been passed to the gateway, either through an explicit setCallbackWithSessionID() invocation, or via the return of the reportNotification() method.

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

Case a:

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

Case b:

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

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

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

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

Parameters

callReference : in TpMultiPartyCallIdentifier

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

callLegReferenceSet : in TpCallLegIdentifierSet

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.

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

notificationInfo : in TpCallNotificationInfo

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

assignmentID : in TpAssignmentID

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

Returns

TpAppMultiPartyCallBack

End of Change in Clause 6.2 End Of Document

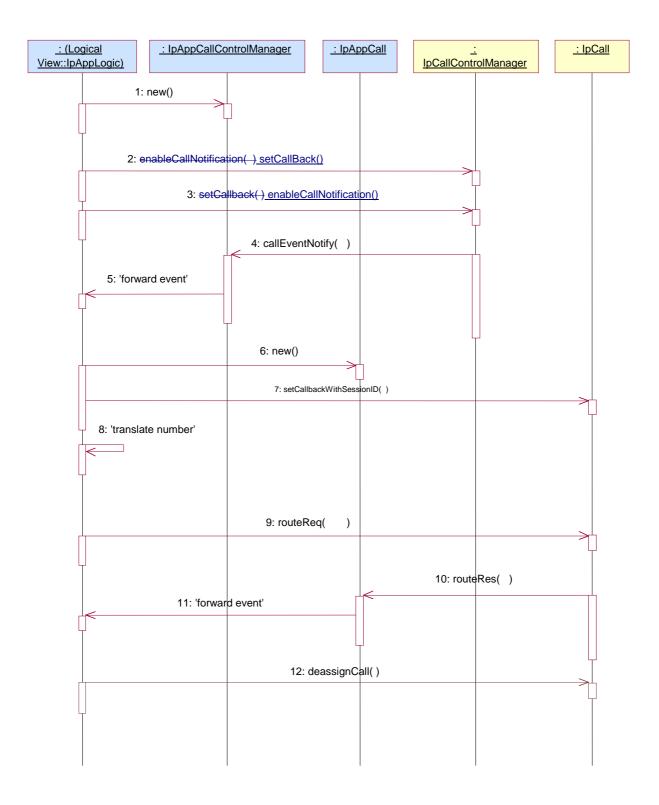
	rlay, ETSI Project OSA, 3GPP TSG_CN WG5) Iona, SPAIN, 01-05 November 2004	N5-040721
	CHANGE REQUEST	CR-Form-v7
^ж 29.198	•04-2 CR 026 # rev - [#] Current version:	6.2.0 [#]
For <u>HELP</u> on usin	this form, see bottom of this page or look at the pop-up text ove	r the X symbols.
Proposed change aff	cts: UICC apps# ME Radio Access Network	Core Network X
Title: ೫ (prrect Behaviour of CallBack sequence and timing	
Source: ೫ (N5 AePONA (Eamonn Murray)	
Work item code: 🛱 🤇	SA1 Date: # 05	5/11/2004
Category: ೫	Release: ೫ R	EL-6
		following releases:
		M Phase 2) lease 1996)
		lease 1997)
		lease 1998) lease 1999)
De		lease 4)
be		lease 5)
	Rel-6 (Rel	lease 6)
Reason for change: Summary of change:	call control services has been reported from the second OSA/ event. In particular the sequence and timing of specifying call subject to different interpretations amongst vendors of applica This was recognised as a major interoperability problem at the OSA/Parlay Interoperability test. As a result, CRs to improve the description of callback behavior introduced during CN5#27 as contributions N5-040338 throug However the resulting specification text that has been produce significant ambiguities regarding the use of the callback function interoperability between application and service implementation callback features, cannot be clearly understood. It is therefore recommended to further clarify the description of callback features such that a clear and common understanding vendors of applications and services.	(Parlay PLUGTEST backs has been ations and services. e second four were gh N5-040342. ed, retains tonality, such that ons, using the of the use of the ig is possible for
	method semantics and correction to existing sequence diagra	m.
Consequences if not approved:	Interoperability cannot be supported, as the existing specification open to multiple, differing, interpretations.	tion shall remain
Clauses affected:	4.6, 6.1, 6.2,	
Clauses allected.	, 0.1, 0.2,	
Other specs affected:	YNXOther core specifications#XTest specificationsXO&M Specifications	
Other comments:	Rel-6 Mirror CR to N5-040718 Rel 4 CR 29.198-04	

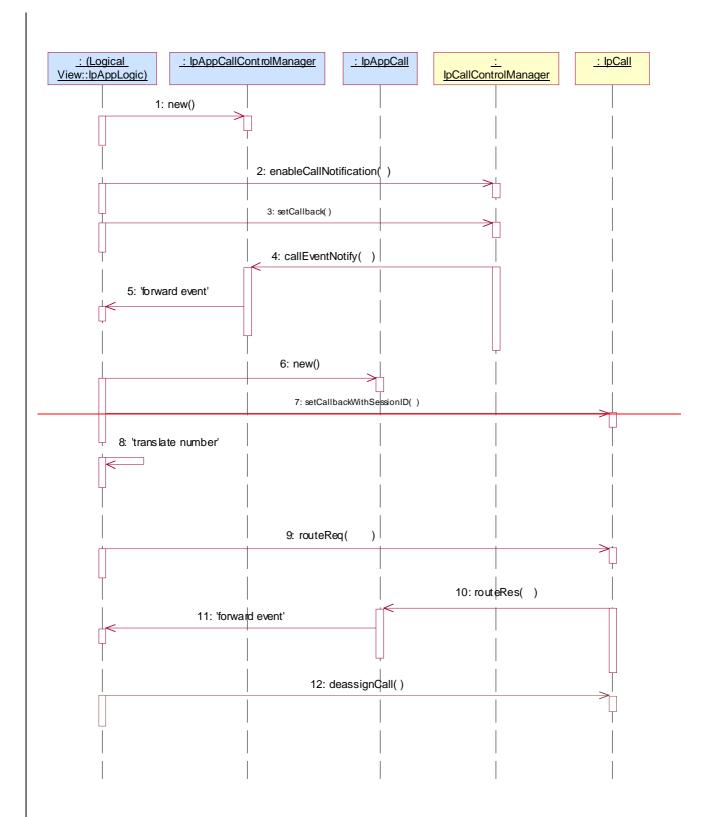
Change in Clause 4.6

4.6 Number Translation 1 (with callbacks)

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

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





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

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

enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification.

3: This message sets the reference of the IpAppCallControlManager object in the CallControlManager. The CallControlManager reports the callEventNotify to referenced object only for enableCallNotifications that do not have a explicit IpAppCallControlManager reference specified in the enableCallNotification. This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a number translation service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria set in message 3, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) is met, other messages (not shown) are used to create the call and associated call leg object.

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

- 5: This message is used to forward message 4 to the IpAppLogic.
- 6: This message is used by the application to create an object implementing the IpAppCall interface.
- 7: This message is used to set the reference to the IpAppCall for this call.
- 8: This message invokes the number translation function.

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

- 10: This message passes the result of the call being answered to its callback object
- 11: This message is used to forward the previous message to the IpAppLogic.

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

End of Change in Clause 4.6

Change in Clause 6.1

6.1 Interface Class IpCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Generic Call Control Service. The generic call control manager interface provides the management functions to the generic call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications.

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

<<Interface>>

IpCallControlManager

createCall (appCall : in IpAppCallRef) : TpCallIdentifier

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

disableCallNotification (assignmentID : in TpAssignmentID) : void

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

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

getCriteria () : TpCallEventCriteriaResultSet

6.1.1 Method createCall()

This method is used to create a new call object.

Call back reference:

An IpAppCallControlManager should already have been passed to the IpCallControlManager, otherwise the call control will not be able to report a callAborted() to the application. The application <u>should_shall</u> invoke setCallback() prior to createCall if it wishes to ensure this.

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

Parameters

appCall : in IpAppCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 Method enableCallNotification()

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

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

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

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

Setting the callback reference:

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

Case a:

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

Case b:

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

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

Set additional callback:

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. See examples in 6.1.1.

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

Parameters

appCallControlManager : in IpAppCallControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified <u>previously</u> 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>></interface>
IpAppCallControlManager
callAborted (callReference : in TpSessionID) : void
callEventNotify (callReference : in TpCallIdentifier, eventInfo : in TpCallEventInfo, assignmentID : in TpAssignmentID) : IpAppCallRef
callNotificationInterrupted () : void
callNotificationContinued () : void
callOverloadEncountered (assignmentID : in TpAssignmentID) : void
callOverloadCeased (assignmentID : in TpAssignmentID) : void

6.2.1 Method callAborted()

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

Parameters

callReference : in TpSessionID

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

6.2.2 Method callEventNotify()

This method notifies the application of the arrival of a call-related event.

If this method is invoked with a monitor mode of P_CALL_MONITOR_MODE_INTERRUPT, then the APL has control of the call. If the APL does nothing with the call (including its associated legs) within a specified time period (the duration of which forms a part of the service level agreement), then the call in the network shall be released and callEnded() shall be invoked, giving a release cause of 102 (Recovery on timer expiry).

Setting the callback reference:

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

Case a:

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

Case b:

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

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 <u>previously</u> by setCallbackWithSessionID(). See example in 4.6

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

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

Parameters

callReference : in TpCallIdentifier

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

eventInfo : in TpCallEventInfo

Specifies data associated with this event.

assignmentID : in TpAssignmentID

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

Returns

IpAppCallRef

End of Change in Clause 6.2 End Of Document

			ETSI Projec SPAIN, 01-(-040722
			CHAI		EQUE	EST				CR-Form-v7
ж	<mark>29.198</mark>	<mark>3-04-3</mark>	CR 033	ж г	ev -	ж	Current vers	sion:	6.3.0	ж
For <mark>HEL</mark>	P on usin	g this for	m, see bottom	of this pag	e or look	at th	e pop-up text	t over	the X syr	nbols.
Proposed cl	hange affe	ects: l	JICC apps೫	M	E 🔜 Ra	idio A	ccess Netwo	rk	Core Ne	etwork X
Title:	ж (Correct B	ehaviour of Ca	allBack seq	uence an	d tim	ing			
Source:	ж (CN5 AeP	ONA (Eamonr	Murray)						
Work item c	ode: 🕱 🤇	OSA1					Date: #	05/	11/2004	
Category:	ж <mark>/</mark>	N.					Release: #	RE	L-6	
0,	Us	se <u>one</u> of a F (cori	the following ca	tegories:			Use <u>one</u> of 2		ollowing rele // Phase 2)	eases:
		A (con	responds to a c		n earlier r	elease		(Rele	ease 1996)	
			lition of feature) ctional modifica		a)		R97 R98		ease 1997)	
			torial modificatio		e)		R90 R99		ease 1998) ease 1999)	
	De		planations of the		gories car	n	Rel-4		ease 4)	
	be	found in	3GPP <u>TR 21.90</u>	<u>0</u> .			Rel-5	•	ease 5)	
							Rel-6	(Rele	ease 6)	
Reason for o		call c even subje This OSA As a introd signi inter callb	nderstandings control service t. In particular ect to different was recognise /Parlay Interop result, CRs to duced during (ever the result ficant ambigui operability bet ack features, o herefore recor ack features s ors of applicat	s has been the sequer interpretati ed as a maj perability ter improve th CN5#27 as ing specific ties regardi ween applic cannot be c nmended to uch that a c	reported loce and ti ons amo or interop st. e descrip contribut ation tex ng the us cation and learly un o further clear and	from iming ngst v perab otion of ions I t that d serv dersto	the second C of specifying vendors of ap ility problem a of callback be N5-040338 th has been pro- he callback f vice impleme bod.	DSA/F callb plicat at the ehavic arough oduce unctic ntatio	Parlay PLU acks has ions and s second our were n N5-0403 d, retains onality, suc ns, using t the use o	JGTEST been services. 442. ch that the f the
Summary of	^r change:		ectly define the od semantics							existing
Consequend not approve			operability car to mutliple, d				existing spec	ificati	on shall re	emain
Clauses affe	ected:	೫ <mark>6.1,</mark> 0	6.2							
		,								
Other specs affected:	;	¥ N ₭ <mark>2 X</mark> X X	Other core s Test specific O&M Specifi	ations	s ¥					
Other comm	ents:	쁐 <mark>Rel-6</mark>	6 Mirror CR to	<mark>N5-040718</mark>	Rel 4 Cl	R 29.	198-04			

Change in Clause 6.1

6.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Multi-party Call Control Service. The multi-party call control manager interface provides the management functions to the multi-party call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications. The action table associated with the STD shows in what state the IpMultiPartyCallControlManager must be if a method can successfully complete. In other words, if the IpMultiPartyCallControlManager is in another state the method will throw an exception immediately.

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

. Interfeces

< <interface>></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</deprecated>
a to all and O antical (duration), in Ta Duration, and charge and in Ta O all and O antical Mark anisms, the star and sig

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

enableNotifications (appCallControlManager : in IpAppMultiPartyCallControlManagerRef) : TpAssignmentID

disableNotifications () : void

getNextNotification (reset : in TpBoolean) : TpNotificationRequestedSetEntry

6.1.1 Method createCall()

This method is used to create a new call object. An IpAppMultiPartyCallControlManager should already have been passed to the IpMultiPartyCallControlManager, otherwise the call control will not be able to report a callAborted() to the application. The application should should should

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

Parameters

appCall : in IpAppMultiPartyCallRef

Specifies the application interface for callbacks from the call created.

Returns

TpMultiPartyCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 Method createNotification()

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

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

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

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

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

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

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

Setting the callback reference:

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

Case a:

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

Case b:

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

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

Setting additional callback:

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

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

Parameters

appCallControlManager : in IpAppMultiPartyCallControlManagerRef

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

notificationRequest : in TpCallNotificationRequest

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

6.1.3 Method destroyNotification()

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

Parameters

assignmentID : in TpAssignmentID

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

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

6.1.4 Method changeNotification()

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

Parameters

assignmentID : in TpAssignmentID

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

notificationRequest : in TpCallNotificationRequest

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

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

6.1.5 Method <<deprecated>> getNotification()

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

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns notificationsRequested: Specifies the notifications that have been requested by the application. An empty set is returned when no notifications exist.

Parameters

No Parameters were identified for this method

Returns

TpNotificationRequestedSet

Raises

TpCommonExceptions

6.1.6 Method setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

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

Parameters

duration : in TpDuration

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

mechanism : in TpCallLoadControlMechanism

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

treatment : in TpCallTreatment

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

addressRange : in TpAddressRange

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

6.1.7 Method enableNotifications()

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

Setting the callback reference:

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

Case a:

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

Case b:

The enableNotifications() with no callback reference ("Null" value) is used where (e.g. due to distributed application logic) the callback reference is provided <u>subsequently previously</u> in a setCallback().<u>If no callback reference has been</u> provided previously to the service, the exception, P NO CALLBACK ADDRESS SET shall be raised.

In case the <u>createNotificationenableNotifications()</u> contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Setting additional Call back:

If the same application requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

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

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

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

Parameters

appCallControlManager : in IpAppMultiPartyCallControlManagerRef

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

Returns

TpAssignmentID

Raises

TpCommonExceptions

End of Change in Clause 6.1

Change in Clause 6.2

6.2 Interface Class IpAppMultiPartyCallControlManager

Inherits from: IpInterface

The Multi-Party call control manager application interface provides the application call control management functions to the Multi-Party call control service.

< <interface>></interface>
IpAppMultiPartyCallControlManager
reportNotification (callReference : in TpMultiPartyCallIdentifier, callLegReferenceSet : in TpCallLegIdentifierSet, notificationInfo : in TpCallNotificationInfo, assignmentID : in TpAssignmentID) : TpAppMultiPartyCallBack
callAborted (callReference : in TpSessionID) : void
managerInterrupted () : void
managerResumed () : void
callOverloadEncountered (assignmentID : in TpAssignmentID) : void
callOverloadCeased (assignmentID : in TpAssignmentID) : void

6.2.1 Method reportNotification()

This method notifies the application of the arrival of a call-related event.

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

Setting the callback reference:

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

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

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

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

Case a:

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

Case b:

The reportNotification() with no call back reference ("Null" value) is used where (e.g. due to distributed application logic) the call back reference is provided <u>subsequently-previously</u> in a setCallbackWithSessionID(). <u>If no callback</u> reference has been provided previously to the service, the exception, P NO CALLBACK ADDRESS SET shall be raised, and no further application invocations related to the call shall be permitted

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

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

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

Parameters

callReference : in TpMultiPartyCallIdentifier

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

callLegReferenceSet : in TpCallLegIdentifierSet

Specifies the set of all call leg references. First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.

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

notificationInfo : in TpCallNotificationInfo

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

assignmentID : in TpAssignmentID

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

Returns

TpAppMultiPartyCallBack

End of Change in Clause 6.2 End Of Document