3GPP TSG CN Plenary Meeting #25 **NP-040358**
08-10 September 2004, Palm Springs, CA, USA

Source: **CN5 (OSA)**

Title: **18 Rel-6 CR 29.198-xy Corrections to support High-Availability (HA)**

Agenda item: **9.7 (OSA Enhancements [OSA3])**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	R	Phase	Subject	Cat	VerCur	Doc-2nd-Level	WI
NP-040358	29.198-03	126	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040660	OSA3
NP-040358	29.198-04-1	014	--	Rel-6	Support High Availability at API Level	F	6.2.0	N5-040646	OSA3
NP-040358	29.198-04-2	023	--	Rel-6	Additional GCC Feature to support HA	C	6.1.0	N5-040611	OSA3
NP-040358	29.198-04-2	024	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040647	OSA3
NP-040358	29.198-04-3	030	--	Rel-6	Additional MPCC Feature to support HA	C	6.2.0	N5-040612	OSA3
NP-040358	29.198-04-3	031	--	Rel-6	Support High Availability at API Level	F	6.2.0	N5-040648	OSA3
NP-040358	29.198-04-4	021	--	Rel-6	Support High Availability at API Level	F	6.2.0	N5-040649	OSA3
NP-040358	29.198-05	057	--	Rel-6	Additional GUI Feature to support HA	C	6.1.0	N5-040613	OSA3
NP-040358	29.198-05	058	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040650	OSA3
NP-040358	29.198-06	029	--	Rel-6	Support High Availability at API Level	F	6.2.0	N5-040651	OSA3
NP-040358	29.198-07	031	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040652	OSA3
NP-040358	29.198-08	036	--	Rel-6	Additional DSC Feature to support HA	C	6.1.0	N5-040614	OSA3
NP-040358	29.198-08	037	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040653	OSA3
NP-040358	29.198-11	032	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040654	OSA3
NP-040358	29.198-12	032	--	Rel-6	Additional Charging Feature to support HA	C	6.1.0	N5-040615	OSA3
NP-040358	29.198-12	033	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040655	OSA3
NP-040358	29.198-13	012	--	Rel-6	Support High Availability at API Level	F	6.2.0	N5-040656	OSA3
NP-040358	29.198-14	024	--	Rel-6	Support High Availability at API Level	F	6.1.0	N5-040657	OSA3

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040611
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-2**	**CR**	**023**	⌘**rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Additional GCC Feature to support HA		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**C**	***Release:*** ⌘	*REL-6*

Use one of the following categories:	*Use one of the following releases:*
F *(correction)*	*2 (GSM Phase 2)*
A *(corresponds to a correction in an earlier release)*	*R96 (Release 1996)*
B *(addition of feature),*	*R97 (Release 1997)*
C *(functional modification of feature)*	*R98 (Release 1998)*
D *(editorial modification)*	*R99 (Release 1999)*
Detailed explanations of the above categories can	*Rel-4 (Release 4)*
be found in 3GPP TR 21.900.	*Rel-5 (Release 5)*
	Rel-6 (Release 6)

Reason for change:	⌘	Current support for high availability for applications and SCSs using the OSA API is ambiguous and incomplete. SCS failure and recovery currently may result in significant additional method invocation, seriously impacting the recovery time and end to end availablility of both application and gateway SCS, and thereby reducing the overall availability of the OSA solution. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	Introduce a new method that allows the failure and recovery of a service to result in a method to be invoked on the application indicating the list of call sessions that have been lost as a result of this failure. The recovery from failure in an SCS can therefore be supported more efficiently thereby improving the end to end availability of the OSA solution.
Consequences if not approved:	⌘	The existing API does not support efficient failure recovery mechanisms required to ensure a highly available end to end OSA deployment, therefore application and gateway implementations shall be required to adopt vendor proprietary solutions to these problems, thereby resulting in loss of interoperability. The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	6.2				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

Change in 6.2

6.2 Interface Class IpAppCallControlManager

Inherits from: IpInterface

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

<<Interface>> IpAppCallControlManager
callAborted (callReference : in TpSessionID) : void callEventNotify (callReference : in TpCallIdentifier, eventInfo : in TpCallEventInfo, assignmentID : in TpAssignmentID) : IpAppCallRef callNotificationInterrupted () : void callNotificationContinued () : void callOverloadEncountered (assignmentID : in TpAssignmentID) : void callOverloadCeased (assignmentID : in TpAssignmentID) : void <<new>>abortMultipleCalls (callReferenceSet : in TpSessionIDSet) : void

6.2.1 Method callAborted()

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

Parameters

callReference : in TpSessionID

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

6.2.2 Method callEventNotify()

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

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

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

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

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

Parameters

`callReference : in TpCallIdentifier`

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

`eventInfo : in TpCallEventInfo`

Specifies data associated with this event.

`assignmentID : in TpAssignmentID`

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

Returns

`IpAppCallRef`

6.2.3 Method callNotificationInterrupted()

This method indicates to the application that all event notifications have been temporarily interrupted (for example, due to faults detected).

Note that more permanent failures are reported via the Framework (integrity management).

Parameters

No Parameters were identified for this method

6.2.4 Method callNotificationContinued()

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

Parameters

No Parameters were identified for this method

6.2.5 Method callOverloadEncountered()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.6 Method callOverloadCeased()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.7 Method abortMultipleCalls()

The service may invoke this method on the IpAppCallControlManager interface to indicate that a number of ongoing call sessions have aborted or terminated abnormally. No further communication will be possible between the application and the calls. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of call sessions have failed. The service shall provide a set of call sessionIDs indicating to the application the call sessions that have aborted. In the case that the service invokes this method and provides an empty set of sessionIDs, this shall be used to indicate that all call sessions previously active on the IpCallControlManager interface have been aborted.

Parameters

`callReferenceSet : in TpSessionIDSet`

Specifies the set of sessionIDs of calls that have aborted or terminated abnormally. The empty set shall be used to indicate that all calls have aborted.

End of Change in 6.2

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040612
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-3**	**CR**	**030**	⌘**rev**	**-**	⌘	Current version:	**6.2.0** ⌘

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

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

Title: ⌘	Additional MPCC Feature to support HA			
Source: ⌘	CN5 AePONA – Eamonn Murray			
Work item code: ⌘	OSA3		***Date:*** ⌘	27/08/2004
Category: ⌘	**C**		***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change: ⌘	Current support for high availability for applications and SCSs using the OSA API is ambiguous and incomplete. SCS failure and recovery currently may result in significant additional method invocation, seriously impacting the recovery time and end to end availablility of both application and gateway SCS, and thereby reducing the overall availability of the OSA solution. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change: ⌘	Introduce a new method that allows the failure and recovery of a service to result in a method to be invoked on the application indicating the list of call sessions that have been lost as a result of this failure. The recovery from failure in an SCS can therefore be supported more efficiently thereby improving the end to end availability of the OSA solution.
Consequences if not approved: ⌘	The existing API does not support efficient failure recovery mechanisms required to ensure a highly available end to end OSA deployment, therefore application and gateway implementations shall be required to adopt vendor proprietary solutions to these problems, thereby resulting in loss of interoperability. The Release 6 stage 1 requirements cannot be addressed.

Clauses affected: ⌘	6.2			
	Y	**N**		
Other specs affected: ⌘		**X**	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	
Other comments: ⌘				

How to create CRs using this form:

Change in 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

<<new>>abortMultipleCalls (callReferenceSet : in TpSessionIDSet) : 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.

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.

Parameters

callReference : in TpMultiPartyCallIdentifier

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

callLegReferenceSet : in TpCallLegIdentifierSet

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

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

notificationInfo : in TpCallNotificationInfo

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

assignmentID : in TpAssignmentID

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

Returns

TpAppMultiPartyCallBack

6.2.2 Method callAborted()

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

Parameters

callReference : in TpSessionID

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

6.2.3 Method managerInterrupted()

This method indicates to the application that event notifications and method invocations have been temporarily interrupted (for example, due to network resources unavailable).

Note that more permanent failures are reported via the Framework (integrity management).

Parameters

No Parameters were identified for this method

6.2.4 Method managerResumed()

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

Parameters

No Parameters were identified for this method

6.2.5 Method callOverloadEncountered()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.6 Method callOverloadCeased()

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

Parameters

`assignmentID : in TpAssignmentID`

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

6.2.7 Method abortMultipleCalls()

The service may invoke this method on the IpAppCallControlManager interface to indicate that a number of ongoing call sessions have aborted or terminated abnormally. No further communication will be possible between the application and the calls. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of call sessions have failed. The service shall provide a set of call sessionIDs indicating to the application the call sessions that have aborted. In the case that the service invokes this method and provides an empty set of sessionIDs, this shall be used to indicate that all call sessions previously active on the IpCallControlManager interface have been aborted.

Parameters

`callReferenceSet : in TpSessionIDSet`

Specifies the set of sessionIDs of calls that have aborted or terminated abnormally. The empty set shall be used to indicate that all calls have aborted.

End of Change in 6.2

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040613**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-05** **CR** **057** ⌘**rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title:	⌘	Additional GUI Feature to support HA		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**C**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current support for high availability for applications and SCSs using the OSA API is ambiguous and incomplete. SCS failure and recovery currently may result in significant additional method invocation, seriously impacting the recovery time and end to end availablility of both application and gateway SCS, and thereby reducing the overall availability of the OSA solution. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	Introduce a new method that allows the failure and recovery of a service to result in a method to be invoked on the application indicating the list of user interaction sessions that have been lost as a result of this failure. The recovery from failure in an SCS can therefore be supported more efficiently thereby improving the end to end availability of the OSA solution.
Consequences if not approved:	⌘	The existing API does not support efficient failure recovery mechanisms required to ensure a highly available end to end OSA deployment, therefore application and gateway implementations shall be required to adopt vendor proprietary solutions to these problems, thereby resulting in loss of interoperability. The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	8.1.2, 11				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

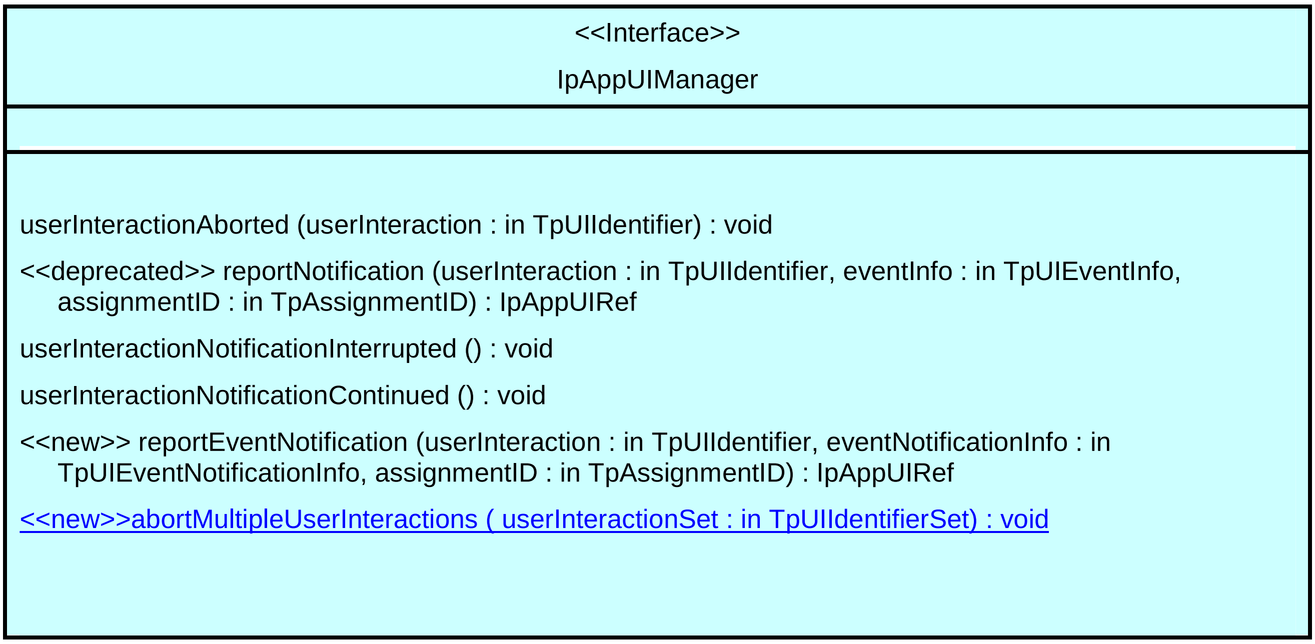
How to create CRs using this form:

Change in 8.1.2

8.1.2 Interface Class IpAppUIManager

Inherits from: IpInterface.

The Generic User Interaction Service manager application interface provides the application callback functions to the Generic User Interaction Service.



<<Interface>>

IpAppUIManager

userInteractionAborted (userInteraction : in TpUIIdentifier) : void

<<deprecated>> reportNotification (userInteraction : in TpUIIdentifier, eventInfo : in TpUIEventInfo, assignmentID : in TpAssignmentID) : IpAppUIRef

userInteractionNotificationInterrupted () : void

userInteractionNotificationContinued () : void

<<new>> reportEventNotification (userInteraction : in TpUIIdentifier, eventNotificationInfo : in TpUIEventNotificationInfo, assignmentID : in TpAssignmentID) : IpAppUIRef

<<new>>abortMultipleUserInteractions (userInteractionSet : in TpUIIdentifierSet) : void

8.1.2.1 Method userInteractionAborted()

This method indicates to the application that the User Interaction service instance has terminated or closed abnormally. No further communication will be possible between the User Interaction service instance and application.

Parameters

`userInteraction : in TpUIIdentifier`

Specifies the interface and sessionID of the user interaction service that has terminated.

8.1.2.2 Method <<deprecated>> reportNotification()

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

This method notifies the application of an occurred network event which matches the criteria installed by the createNotification method.

Returns: appUI

Specifies a reference to the application interface, which implements the callback interface for the new user interaction.

If the application has previously explicitly passed a reference to the IpAppUI interface using a setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

Parameters

`userInteraction : in TpUIIdentifier`

Specifies the reference to the interface and the sessionID to which the notification relates.

`eventInfo : in TpUIEventInfo`

Specifies data associated with this event.

`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

`IpAppUIRef`

8.1.2.3 Method userInteractionNotificationInterrupted()

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

Parameters

No Parameters were identified for this method

8.1.2.4 Method userInteractionNotificationContinued()

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

Parameters

No Parameters were identified for this method

8.1.2.5 Method <<new>> reportEventNotification()

This method notifies the application of an occurred network event which matches the criteria installed by the createNotification method.

Returns: appUI

Specifies a reference to the application interface, which implements the callback interface for the new user interaction.

If the application has previously explicitly passed a reference to the IpAppUI interface using a setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

Parameters

`userInteraction : in TpUIIdentifier`

Specifies the reference to the interface and the sessionID to which the notification relates.

`eventNotificationInfo : in TpUIEventNotificationInfo`

Specifies data associated with this event.

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

IpAppUIRef

8.1.2.6 Method abortMultipleUserInteractions()

The service may invoke this method on the IpAppUIManager interface to indicate that a number of ongoing user interaction sessions have aborted or terminated abnormally. No further communication will be possible between the application and the user interaction sessions. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of sessions have failed. The service shall provide a set of TpUIIdentifiers, indicating to the application the interface references and sessionsIDs of the user interaction sessions that have aborted. In the case that the service invokes this method and provides an empty set of TpUIIdentifiers, this shall be used to indicate that all user interaction sessions previously active on the IpUIManager interface have been aborted.

Parameters

userInteractionSet : in TpUIIdentifierSet

Specifies the set of interfaces and sessionIDs of the user interaction sessions that have aborted or terminated abnormally. The empty set shall be used to indicate that all user interactions have aborted.

End of Change in 8.1.2

Change in 11

11.16 TpUIIdentifier

Defines the Sequence of Data Elements that unambiguously specify the UI object

Structure Element Name	Structure Element Type	Structure Element Description
UIRef	IpUIRef	This element specifies the interface reference for the UI object.
UserInteractionSessionID	TpSessionID	This element specifies the User Interaction session ID.

11.17 TpUIIdentifierSet

Defines a Numbered Set of Data Elements of TpUIIdentifier.

End of Change in 11

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040614**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **036** ⌘ **rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title:	⌘	Additional DSC Feature to support HA		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**C**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current support for high availability for applications and SCSs using the OSA API is ambiguous and incomplete. SCS failure and recovery currently may result in significant additional method invocation, seriously impacting the recovery time and end to end availablility of both application and gateway SCS, and thereby reducing the overall availability of the OSA solution. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	Introduce a new method that allows the failure and recovery of a service to result in a method to be invoked on the application indicating the list of data sessions that have been lost as a result of this failure. The recovery from failure in an SCS can therefore be supported more efficiently thereby improving the end to end availability of the OSA solution.
Consequences if not approved:	⌘	The existing API does not support efficient failure recovery mechanisms required to ensure a highly available end to end OSA deployment, therefore application and gateway implementations shall be required to adopt vendor proprietary solutions to these problems, thereby resulting in loss of interoperability. The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	8.2			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:

Change in 8.2

8.2 Interface Class IpAppDataSessionControlManager

Inherits from: IpInterface.

The data session control manager application interface provides the application data session control management functions to the data session control SCF.

<<Interface>> IpAppDataSessionControlManager
dataSessionAborted (dataSession : in TpSessionID) : void reportNotification (dataSessionReference : in TpDataSessionIdentifier, eventInfo : in TpDataSessionEventInfo, assignmentID : in TpAssignmentID) : IpAppDataSessionRef dataSessionNotificationContinued () : void dataSessionNotificationInterrupted () : void <<new>>abortMultipleDataSessions (dataSessionSet : in TpSessionIDSet) : void

8.2.1 Method dataSessionAborted()

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

Parameters

dataSession : in TpSessionID

Specifies the session ID of the data session that has aborted or terminated abnormally.

8.2.2 Method reportNotification()

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

If this method is invoked with a monitor mode of P_DATA_SESSION_MONITOR_MODE_INTERRUPT, then the application has control of the data session. If the application does nothing with the data session within a specified time period (the duration of which forms a part of the service level agreement), then the data session in the network shall be released and dataSessionFaultDetected() shall be invoked, giving a fault code of P_DATA_SESSION_TIMEOUT_ON_INTERRUPT.

Returns appDataSession : Specifies a reference to the application object which implements the callback interface for the new data session. If the application has previously explicitly passed a reference to the IpAppDataSession 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.

Parameters

`dataSessionReference : in TpDataSessionIdentifier`

Specifies the session ID and the reference to the Data Session object 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.

`eventInfo : in TpDataSessionEventInfo`

Specifies data associated with this event. This data includes the destination address provided by the end-user and the quality of service requested or negotiated for the data session.

`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

`IpAppDataSessionRef`

8.2.3 Method dataSessionNotificationContinued()

This method indicates to the application that all event notifications are resumed.

Parameters

No Parameters were identified for this method

8.2.4 Method dataSessionNotificationInterrupted()

This method indicates to the application that event notifications will no longer be sent (for example, due to faults detected).

Parameters

No Parameters were identified for this method

8.2.5 Method abortMultipleDataSessions()

The service may invoke this method on the IpAppDataSessionControlManager interface to indicate that a number of ongoing data sessions have aborted or terminated abnormally. No further communication will be possible between the application and the data sessions. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of data sessions have failed. The service shall provide a set of data sessionIDs indicating to the application the data sessions that have aborted. In the case that the service invokes this method and provides an empty set of sessionIDs, this shall be used to indicate that all data sessions previously active on the IpDataSessionControlManager interface have been aborted.

Parameters

`dataSessionSet : in TpSessionIDSet`

Specifies the set of sessionIDs of data sessions that have aborted or terminated abnormally. The empty set shall be used to indicate that all data sessions have aborted.

End of Change in 8.2

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040615
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-12**	**CR**	**032**	⌘**rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Additional Charging Feature to support HA		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**C**	***Release:*** ⌘	*REL-6*

Use *one* *of the following categories:*
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use *one* *of the following releases:*

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current support for high availability for applications and SCSs using the OSA API is ambiguous and incomplete. SCS failure and recovery currently may result in significant additional method invocation, seriously impacting the recovery time and end to end availablility of both application and gateway SCS, and thereby reducing the overall availability of the OSA solution. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	Introduce a new method that allows the failure and recovery of a service to result in a method to be invoked on the application indicating the list of charging sessions that have been lost as a result of this failure. The recovery from failure in an SCS can therefore be supported more efficiently thereby improving the end to end availability of the OSA solution.
Consequences if not approved:	⌘	The existing API does not support efficient failure recovery mechanisms required to ensure a highly available end to end OSA deployment, therefore application and gateway implementations shall be required to adopt vendor proprietary solutions to these problems, thereby resulting in loss of interoperability. The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	8.2				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

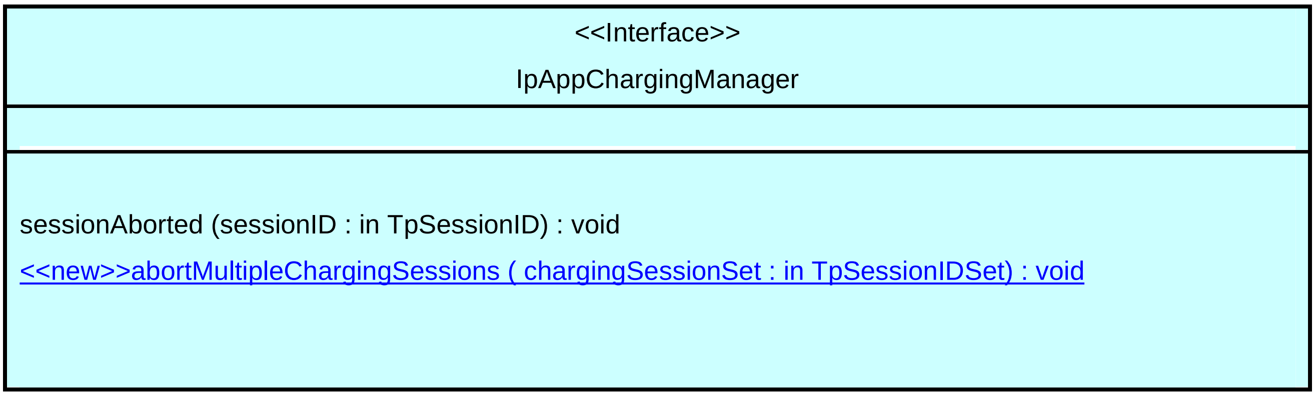
How to create CRs using this form:

Change in 8.2

8.2 Interface Class IpAppChargingManager

Inherits from: IpInterface.

This interface is the manager application interface for the Charging Service. The Charging manager interface provides the application Charging Session Management functions to the charging service.



<<Interface>>
IpAppChargingManager
sessionAborted (sessionID : in TpSessionID) : void
<<new>>abortMultipleChargingSessions (chargingSessionSet : in TpSessionIDSet) : void

8.2.1 Method sessionAborted()

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

Parameters

`sessionID : in TpSessionID`

Specifies the sessionID of the charging session that has aborted or terminated abnormally.

8.2.2 Method abortMultipleChargingSessions()

The service may invoke this method on the IpAppChargingManager interface to indicate that a number of ongoing charging sessions have aborted or terminated abnormally. No further communication will be possible between the application and the charging sessions. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of charging sessions have failed. The service shall provide a set of charging sessionIDs indicating to the application the charging sessions that have aborted. In the case that the service invokes this method and provides an empty set of sessionIDs, this shall be used to indicate that all charging sessions previously active on the IpChargingManager interface have been aborted.

Parameters

`chargingSessionSet : in TpSessionIDSet`

Specifies the set of sessionIDs of charging sessions that have aborted or terminated abnormally. The empty set shall be used to indicate that all charging sessions have aborted.

End of Change in 8.2

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040646**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-1 CR 014** ⌘ **rev -** ⌘ Current version: **6.2.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the CallControl API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	5.4.1				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

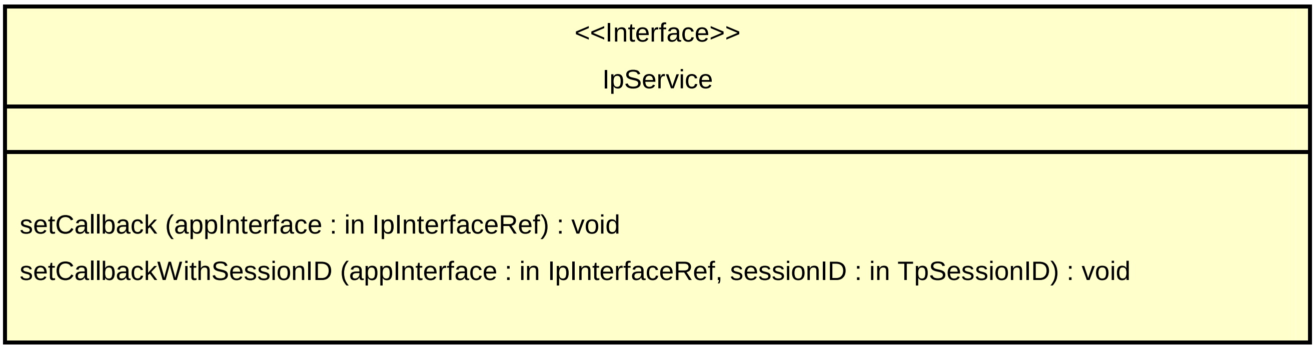
How to create CRs using this form:

Change in 5.4.1

5.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.


<<Interface>>
IpService
setCallback (appInterface : in IpInterfaceRef) : void
setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

5.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

5.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.~~.~~

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE

End of Change in 5.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040647**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-2** **CR** **024** ⌘ **rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the GCC API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the enableCallNotification and disableCallNotification methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	6.1				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

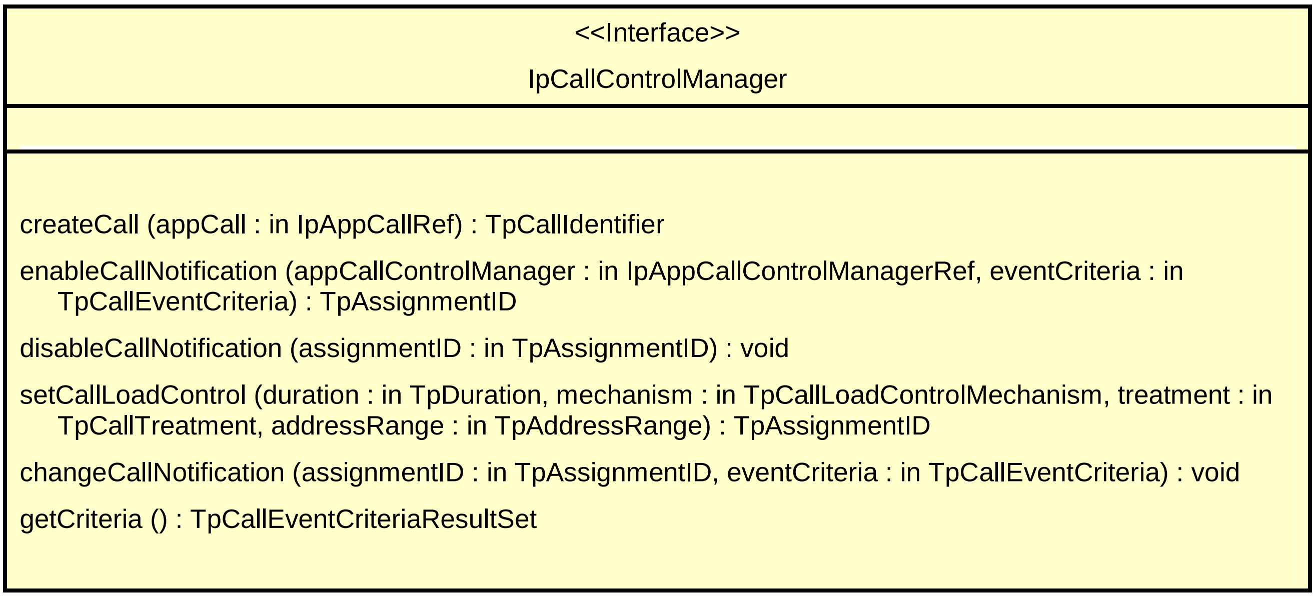
Change in 6.1

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

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

Parameters

`appCall : in IpAppCallRef`

Specifies the application interface for callbacks from the call created.

Returns

`TpCallIdentifier`

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

6.1.2 Method enableCallNotification()

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

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

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

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

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

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

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

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

Parameters

`appCallControlManager : in IpAppCallControlManagerRef`

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

`eventCriteria : in TpCallEventCriteria`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE`

6.1.3 Method disableCallNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

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

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID`

End of Change in 6.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040648**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3** **CR** **031** ⌘ **rev** **-** ⌘ Current version: **6.2.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level			
Source:	⌘	CN5 AePONA – Eamonn Murray			
Work item code:	⌘	OSA3	***Date:***	⌘	02/09/2004
Category:	⌘	**F**	***Release:***	⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the MPCC API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the createNotification and enableNotifications methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	6.1

Other specs affected: ⌘

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

Other comments:	⌘	

How to create CRs using this form:

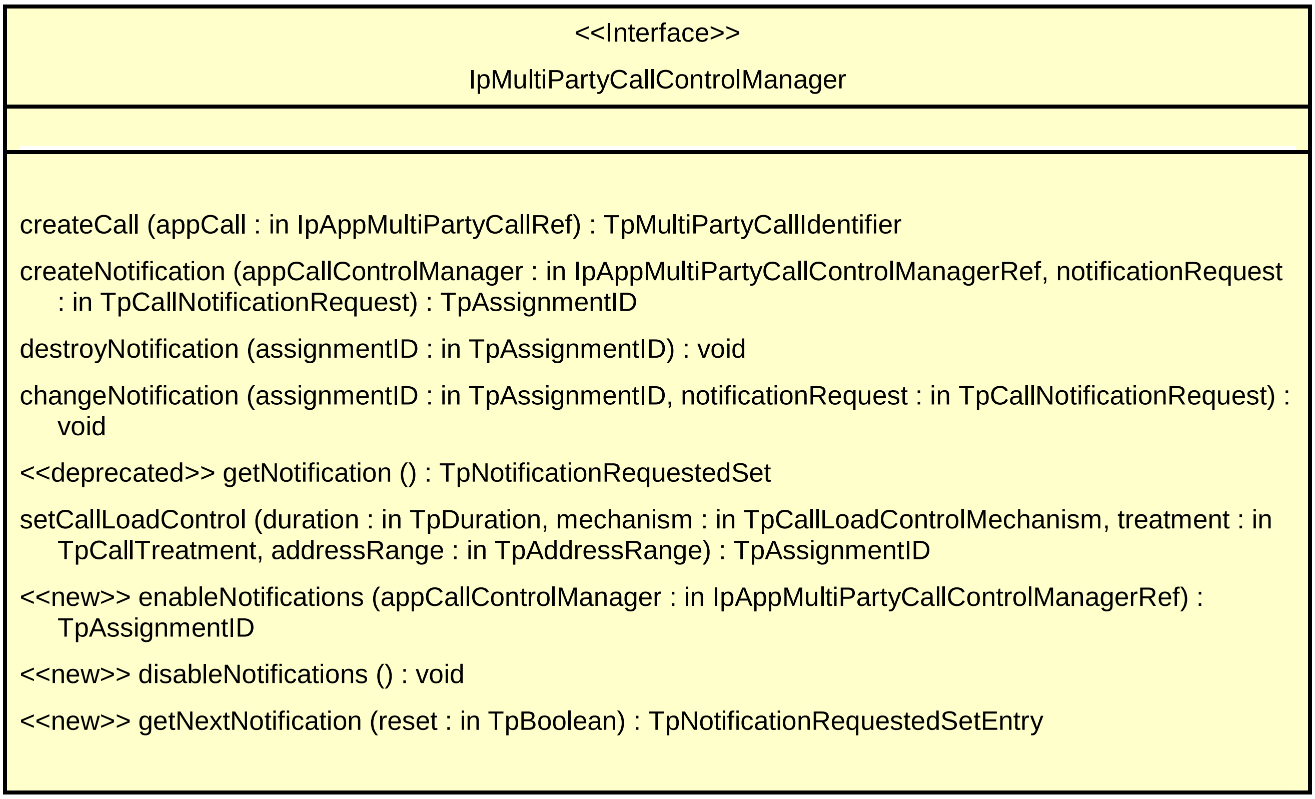
Change in 6.1

6.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

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

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



<<Interface>>

IpMultiPartyCallControlManager

createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier

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

destroyNotification (assignmentID : in TpAssignmentID) : void

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

<<deprecated>> getNotification () : TpNotificationRequestedSet

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

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

<<new>> disableNotifications () : void

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

6.1.1 Method createCall()

This method is used to create a new call object. An IpAppMultiPartyCallControlManager should already have been passed to the IpMultiPartyCallControlManager,

otherwise the call control will not be able to report a callAborted() to the application (the application should invoke setCallback() if it wishes to ensure this).

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

Parameters

`appCall : in IpAppMultiPartyCallRef`

Specifies the application interface for callbacks from the call created.

Returns

TpMultiPartyCallIdentifier

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

6.1.2 Method createNotification()

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

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

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

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

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

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

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

~~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.~~

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

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

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

Parameters

`appCallControlManager : in IpAppMultiPartyCallControlManagerRef`

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

`notificationRequest : in TpCallNotificationRequest`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE`

6.1.3 Method destroyNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

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

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID`

6.1.4 Method changeNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

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

`notificationRequest : in TpCallNotificationRequest`

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

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

6.1.5 Method <<deprecated>> getNotification()

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

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

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

Parameters

No Parameters were identified for this method

Returns

`TpNotificationRequestedSet`

Raises

`TpCommonExceptions`

6.1.6 Method setCallLoadControl()

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

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

Parameters

`duration : in TpDuration`

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

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

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

A duration of -2 indicates the network default duration.

`mechanism : in TpCallLoadControlMechanism`

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

`treatment : in TpCallTreatment`

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

`addressRange : in TpAddressRange`

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN

6.1.7 Method <<new>> enableNotifications()

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

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

If the same application invokes this method multiple times with different IpAppMultiPartyCallControlManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The ~~gateway~~SCS shall use the most recent callback interface provided by the application using this method.~~The gateway shall use multiple callback interfaces in the order that they are provided by the application using this method.~~ In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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 via the setCallback() method.

Returns

TpAssignmentID

Raises

TpCommonExceptions

6.1.8 Method <<new>> disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

6.1.9 Method <<new>> getNextNotification()

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

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

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

Parameters

reset : in TpBoolean

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

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

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

Returns

TpNotificationRequestedSetEntry

Raises

TpCommonExceptions

End of Change in 6.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040649
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-4 CR 021** ⌘ **rev -** ⌘ Current version: **6.2.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the MMCC API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the createMediaNotification method has been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	6.1				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

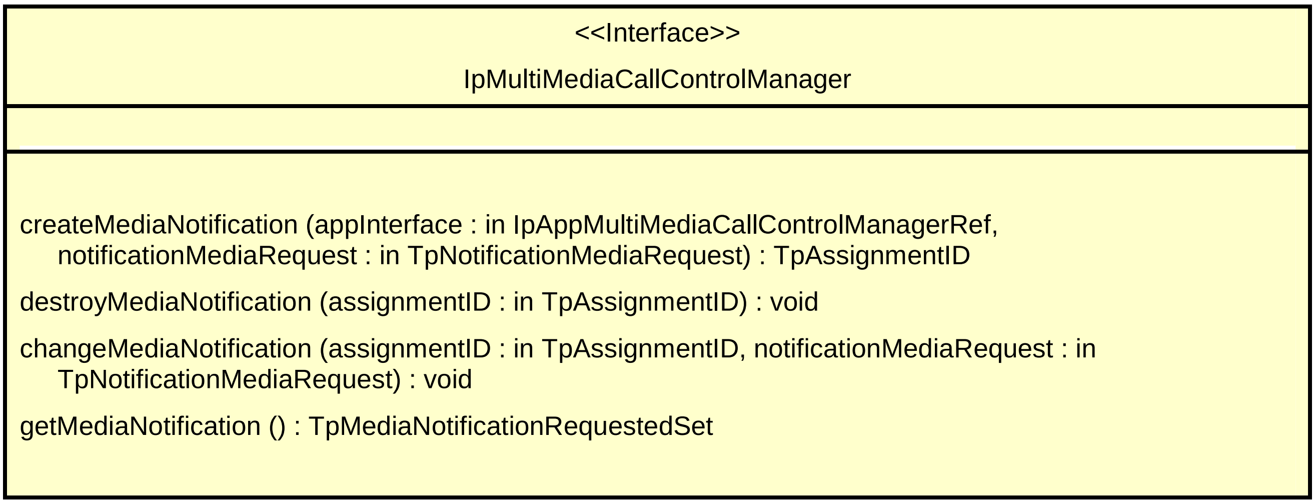
How to create CRs using this form:

Change in 6.1

6.1 Interface Class IpMultiMediaCallControlManager

Inherits from: IpMultiPartyCallControlManager

The Multi Media Call Control Manager is the factory interface for creating multimedia calls. The multi-media call control manager interface provides the management functions to the multi-media call control service. The application programmer can use this interface to create, destroy, change and get media stream related notifications.
This interface shall be implemented by a Multi Media Call Control SCF. As a minimum requirement the createMediaNotification() and destroyMediaNotification() methods shall be implemented. The minimum required methods from IpMultiPartyCallControlManager are also required.



<<Interface>>

IpMultiMediaCallControlManager

createMediaNotification (appInterface : in IpAppMultiMediaCallControlManagerRef, notificationMediaRequest : in TpNotificationMediaRequest) : TpAssignmentID

destroyMediaNotification (assignmentID : in TpAssignmentID) : void

changeMediaNotification (assignmentID : in TpAssignmentID, notificationMediaRequest : in TpNotificationMediaRequest) : void

getMediaNotification () : TpMediaNotificationRequestedSet

6.1.1 Method createMediaNotification()

This method is used to create media stream notifications so that events can be sent to the application.

This applies both to callsetup media (e.g., SIP initial INVITE or H.323 with faststart) and for media setup during the call.

This is the first step an application has to do to get initial notifications of media streams happening in the network. When such an event happens, the application will be informed by reportMediaNotification(). In case the application is interested in other events during the context of a particular call session it has to use the mediaStreamMonitorReq() method on the Multi-Media call leg object.

The createMediaNotification method is purely intended for applications to indicate their interest to be notified when certain media stream events take place. It is possible to subscribe to a certain media stream 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 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.~~

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

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

Returns assignmentID: Specifies the ID assigned by the multi-media call control manager interface for this newly-created notification.

Parameters

appInterface : in IpAppMultiMediaCallControlManagerRef

Specifies a reference to the application interface, which is used for callbacks.

notificationMediaRequest : in TpNotificationMediaRequest

The mediaMonitorMode is a parameter of TpMediaStreamRequest and can be in interrupt or in notify mode. If in interrupt mode the application has to specify which media streams are allowed by calling mediaStreamAllow on the callLeg.

The notificationMediaRequest parameter specifies the event specific criteria used by the application to define the event required. This is the media portion of the criteria. Only events that meet the notificationMediaRequest are reported.

Individual addresses or address ranges may be specified for the destination and/or origination.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE

6.1.2 Method destroyMediaNotification()

This method is used by the application to disable Multi Media Channel notifications

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the Multi Media call control manager interface when the previous enableMediaNotification was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised.

Raises

TpCommonExceptions

6.1.3 Method changeMediaNotification()

This method is used by the application to change the event criteria introduced with createMediaNotification. 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-media call control manager interface for the media stream notification. If two callbacks have been registered under this assignment ID both of them will be disabled.

`notificationMediaRequest : in TpNotificationMediaRequest`

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`

6.1.4 Method getMediaNotification()

This method is used by the application to query the event criteria set with createMediaNotification or changeMediaNotification.

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

Parameters

No Parameters were identified for this method

Returns

`TpMediaNotificationRequestedSet`

Raises

`TpCommonExceptions`

End of Change in 6.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040650
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-05**	**CR**	**058**	⌘**rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the User Interaction API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack, setCallBackWithSessionID, createNotification and enableNotifications methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1, 8.1.1				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

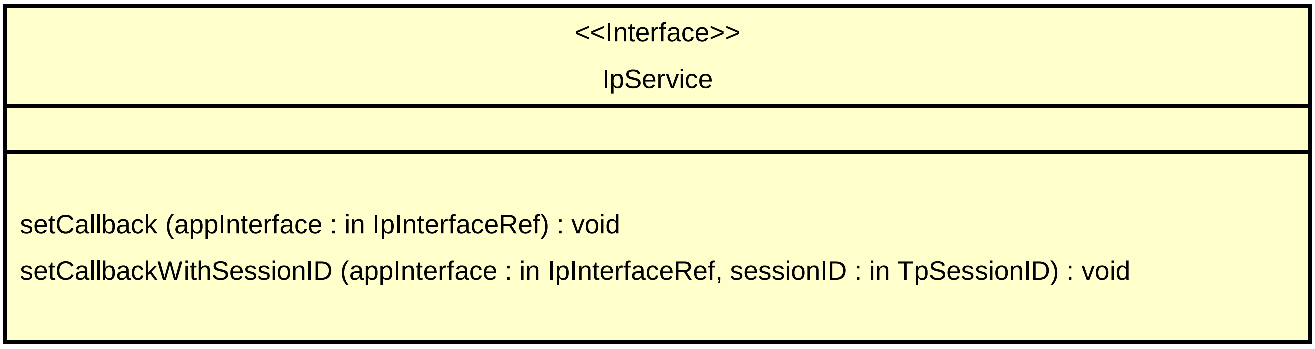
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>
IpService
setCallback (appInterface : in IpInterfaceRef) : void
setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

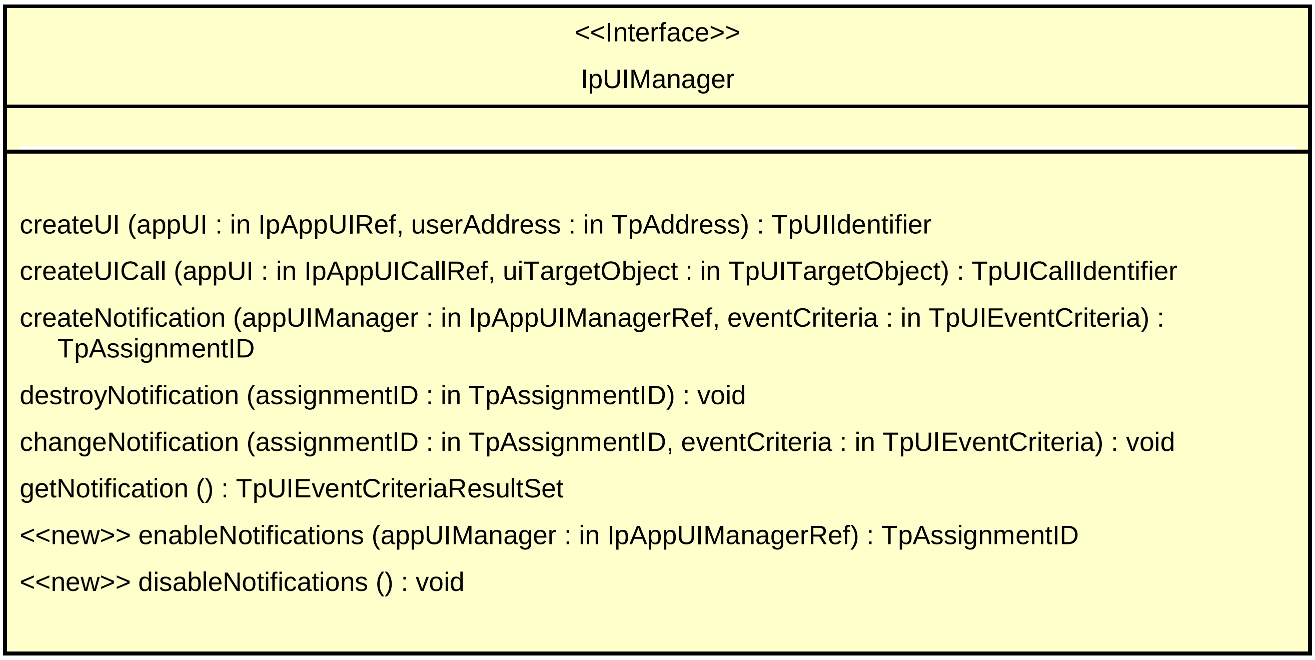
End of Change in 7.4.1

Change in 8.1.1

8.1.1 Interface Class IpUIManager

Inherits from: IpService.

This interface is the 'service manager' interface for the Generic User Interaction Service and provides the management functions to the Generic User Interaction Service.
This interface shall be implemented by a Generic User Interaction SCF. The createUI() method, or the createUICall() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications() and disableNotifications() methods shall be implemented as a minimum requirement.



<<Interface>>

IpUIManager

createUI (appUI : in IpAppUIRef, userAddress : in TpAddress) : TpUIIdentifier

createUICall (appUI : in IpAppUICallRef, uiTargetObject : in TpUITargetObject) : TpUICallIdentifier

createNotification (appUIManager : in IpAppUIManagerRef, eventCriteria : in TpUIEventCriteria) : TpAssignmentID

destroyNotification (assignmentID : in TpAssignmentID) : void

changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpUIEventCriteria) : void

getNotification () : TpUIEventCriteriaResultSet

<<new>> enableNotifications (appUIManager : in IpAppUIManagerRef) : TpAssignmentID

<<new>> disableNotifications () : void

8.1.1.1 Method createUI()

This method is used to create a new user interaction object for non-call related purposes

Results: userInteraction

Specifies the interface and sessionID of the user interaction created.

Parameters

`appUI : in IpAppUIRef`

Specifies the application interface for callbacks from the user interaction created.

`userAddress : in TpAddress`

Indicates the end-user with whom to interact.

Returns

`TpUIIdentifier`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_INTERFACE_TYPE`

8.1.1.2 Method createUICall()

This method is used to create a new user interaction object for call related purposes.

The user interaction can take place to the specified party or to all parties in a call. Note that for certain implementation user interaction can only be performed towards the controlling call party, which shall be the only party in the call.

Returns: userInteraction

Specifies the interface and sessionID of the user interaction created.

Parameters

`appUI : in IpAppUICallRef`

Specifies the application interface for callbacks from the user interaction created.

`uiTargetObject : in TpUITargetObject`

Specifies the object on which to perform the user interaction. This can either be a Call, Multi-party Call or call leg object.

Returns

`TpUICallIdentifier`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_INTERFACE_TYPE`

8.1.1.3 Method createNotification()

This method is used by the application to install specified notification criteria, for which the reporting is implicitly activated. If some application already requested notifications with criteria that overlap the specified criteria, or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA.

The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same servicecode is used.

~~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. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.~~

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction manager interface for this newly installed notification criteria.

Parameters

appUIManager : in IpAppUIManagerRef

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

eventCriteria : in TpUIEventCriteria

Specifies the event specific criteria used by the application to define the event required, like user address and service code.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE

8.1.1.4 Method destroyNotification()

This method is used by the application to destroy previously installed notification criteria via the createNotification method.

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the generic user interaction manager interface when the previous createNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENT_ID.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

8.1.1.5 Method changeNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the manager interface for the event notification.

`eventCriteria : in TpUIEventCriteria`

Specifies the new set of event 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`

8.1.1.6 Method getNotification()

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

Returns: eventCriteria

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

Parameters

No Parameters were identified for this method

Returns

`TpUIEventCriteriaResultSet`

Raises

`TpCommonExceptions`

8.1.1.7 Method <<new>> enableNotifications()

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

~~If the same application requests to enable notifications for a second time with a different IpAppUIManager 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.~~

If the same application invokes this method multiple times with different IpAppUIManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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.

Parameters

`appUIManager : in IpAppUIManagerRef`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

8.1.1.8 Method <<new>> disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

End of Change in 8.1.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040651**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-06**	**CR**	**029**	⌘**rev**	**-**	⌘	Current version:	**6.2.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Mobility API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID, methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

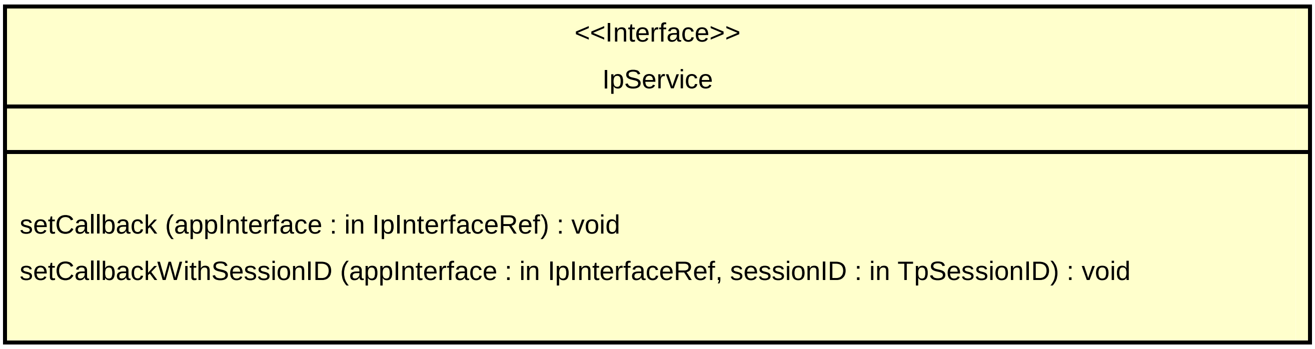
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface : in IpInterfaceRef) : void

setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE

End of Change in 7.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040652**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-07**	**CR**	**031**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level	
Source:	⌘	CN5 AePONA – Eamonn Murray	
Work item code:	⌘	OSA3	***Date:*** ⌘ 02/09/2004
Category:	⌘	**F** *Use one of the following categories:* ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.	***Release:*** ⌘ *REL-6* *Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Terminal Capabilities API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID, methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

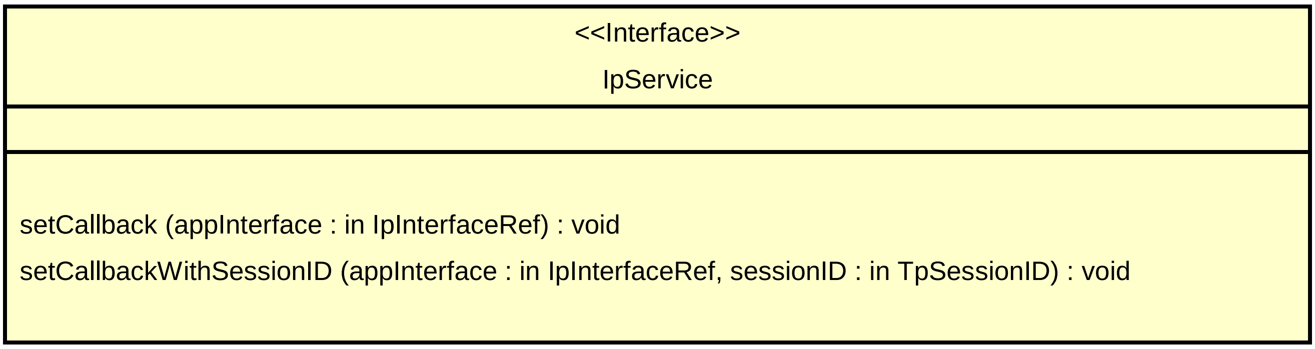
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.


<<Interface>>
IpService
setCallback (appInterface : in IpInterfaceRef) : void
setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

End of Change in 7.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040653**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **037** ⌘**rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Data Session API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack, setCallBackWithSessionID, createNotification, createNotifications and enableNotifications methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1, 8.4			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

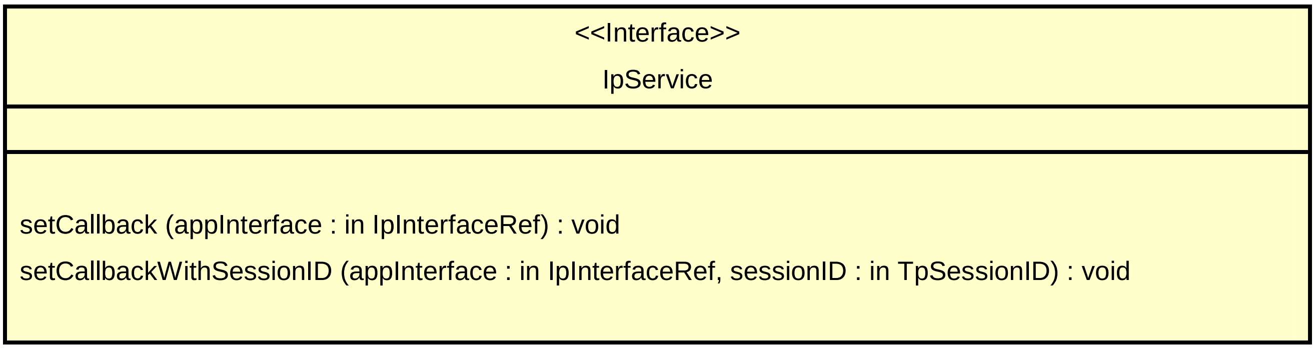
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface : in IpInterfaceRef) : void

setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE

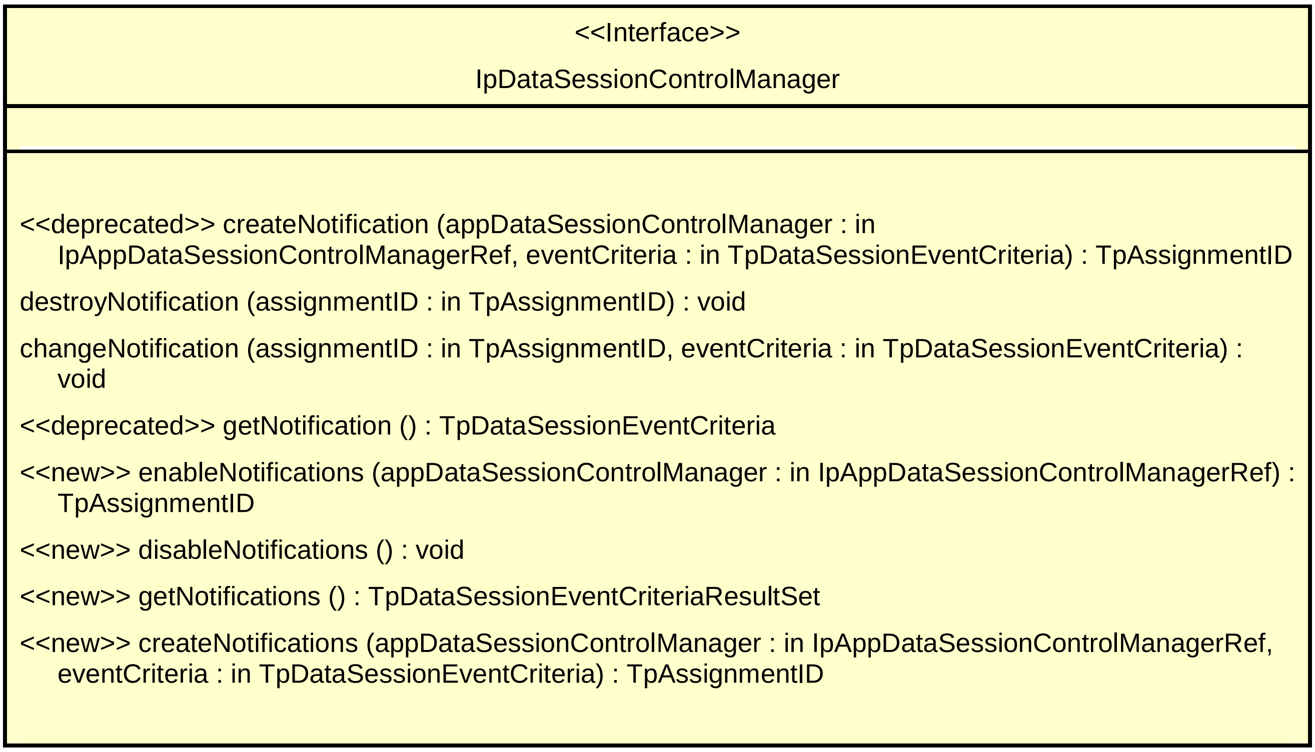
End of Change in 7.4.1

Change in 8.4

8.4 Interface Class IpDataSessionControlManager

Inherits from: IpService.

This interface is the 'SCF manager' interface for Data Session Control. This interface shall be implemented by a Data Session Control SCF. As a minimum requirement, the createNotifications() and destroyNotification(), or the enableNotifications() and disableNotifications() methods shall be implemented.



<<Interface>>

IpDataSessionControlManager

<<deprecated>> createNotification (appDataSessionControlManager : in IpAppDataSessionControlManagerRef, eventCriteria : in TpDataSessionEventCriteria) : TpAssignmentID

destroyNotification (assignmentID : in TpAssignmentID) : void

changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpDataSessionEventCriteria) : void

<<deprecated>> getNotification () : TpDataSessionEventCriteria

<<new>> enableNotifications (appDataSessionControlManager : in IpAppDataSessionControlManagerRef) : TpAssignmentID

<<new>> disableNotifications () : void

<<new>> getNotifications () : TpDataSessionEventCriteriaResultSet

<<new>> createNotifications (appDataSessionControlManager : in IpAppDataSessionControlManagerRef, eventCriteria : in TpDataSessionEventCriteria) : TpAssignmentID

8.4.1 Method <<deprecated>> createNotification()

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

This method is used to enable data session 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 data session 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 data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session 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 data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap 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 give control of a data session. Only one application can place an interrupt request if the criteria overlaps.

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

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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 Data Session Manager object for this newly-enabled event notification.

Parameters

`appDataSessionControlManager : in IpAppDataSessionControlManagerRef`

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

`eventCriteria : in TpDataSessionEventCriteria`

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

8.4.2 Method destroyNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

Specifies the assignment ID given by the data session manager object when the previous createNotification() was done.

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID`

8.4.3 Method changeNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the manager interface for the event notification.

`eventCriteria : in TpDataSessionEventCriteria`

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

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

8.4.4 Method <<deprecated>> getNotification()

This method is deprecated and its use is discouraged. It will be removed in a later release. It is replaced with getNotifications.

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

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

Parameters

No Parameters were identified for this method

Returns

TpDataSessionEventCriteria

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE

8.4.5 Method <<new>> enableNotifications()

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

~~If the same application requests to enable notifications for a second time with a different IpAppDataSessionControlManager 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.~~

If the same application invokes this method multiple times with different IpAppDataSessionControlManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

8.4.6 Method <<new>> disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

8.4.7 Method <<new>> getNotifications()

This method replaces getNotification().

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

Returns eventCriteria: the list of event criteria for the notifications requested by the application. If there is no information to return (e.g. no notifications requested by the application), an empty set (zero length) is returned.

Parameters

No Parameters were identified for this method

Returns

`TpDataSessionEventCriteriaResultSet`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE`

8.4.8 Method <<new>> createNotifications()

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

This method is used to enable data session 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 data session 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 data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session 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 data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap 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 give control of a data session. Only one application can place an interrupt request if the criteria overlaps.

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

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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 Data Session Manager object for this newly-enabled event notification.

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

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

eventCriteria : in TpDataSessionEventCriteria

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_INVALID_INTERFACE_TYPE

End of Change in 8.4

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040654**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-11 CR 032** ⌘**rev** **-** ⌘ Current version: **6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Account Management API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack, setCallBackWithSessionID, createNotification and enableNotifications methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1, 8.1				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

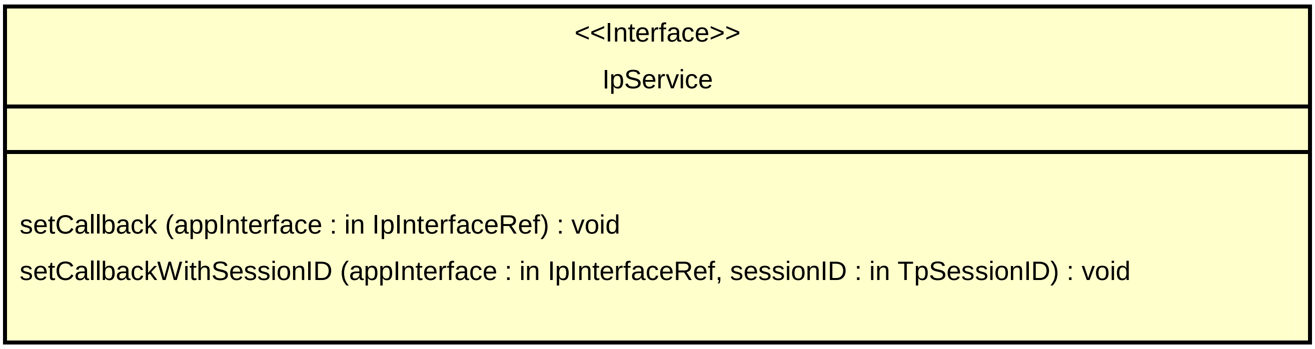
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface : in IpInterfaceRef) : void

setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

End of Change in 7.4.1

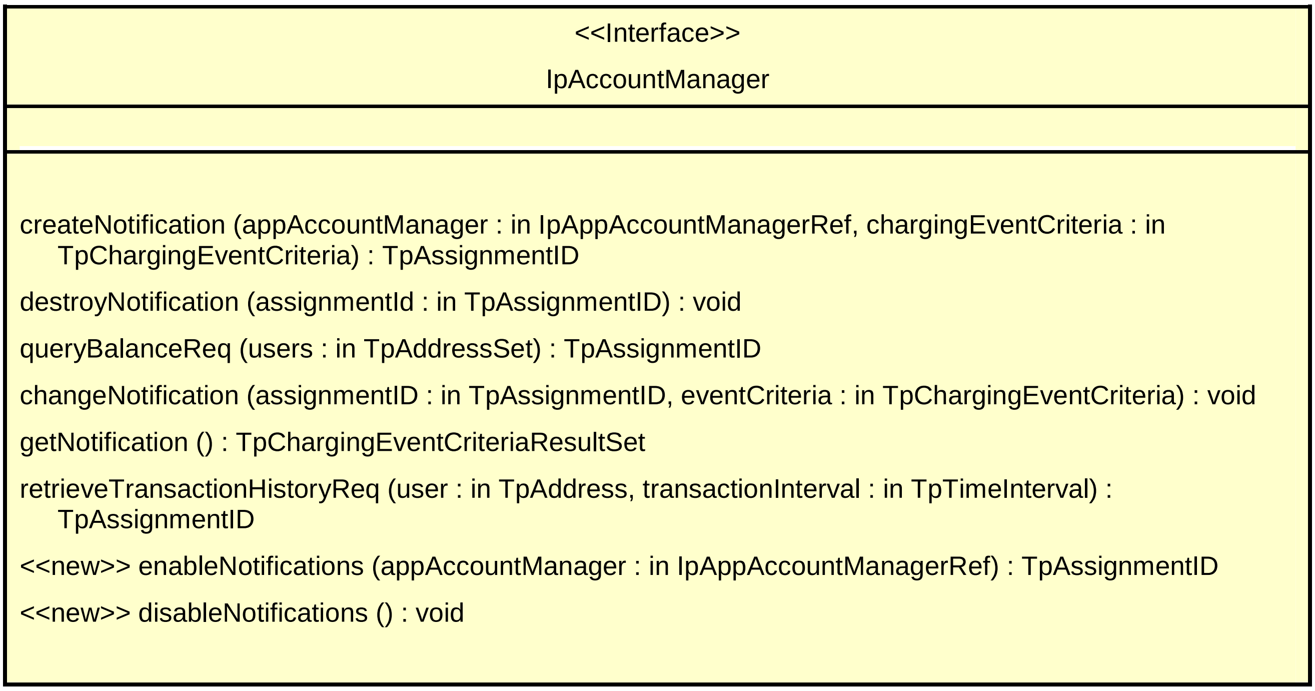
Change in 8.1

8.1 Interface Class IpAccountManager

Inherits from: IpService.

The account manager interface provides methods for monitoring accounts. Applications can use this interface to enable or disable charging-related event notifications and to query account balances.

This interface shall be implemented by an Account Management SCF. The queryBalanceReq() method, or the retrieveTransactionHistoryReq() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications and disableNotifications methods shall be implemented as a minimum requirement.



<<Interface>>

IpAccountManager

createNotification (appAccountManager : in IpAppAccountManagerRef, chargingEventCriteria : in TpChargingEventCriteria) : TpAssignmentID

destroyNotification (assignmentId : in TpAssignmentID) : void

queryBalanceReq (users : in TpAddressSet) : TpAssignmentID

changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpChargingEventCriteria) : void

getNotification () : TpChargingEventCriteriaResultSet

retrieveTransactionHistoryReq (user : in TpAddress, transactionInterval : in TpTimeInterval) : TpAssignmentID

<<new>> enableNotifications (appAccountManager : in IpAppAccountManagerRef) : TpAssignmentID

<<new>> disableNotifications () : void

8.1.1 Method createNotification()

This method is used by the application to enable charging event notifications to be sent to the application.

~~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.~~

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

In case the ~~enableCallNotification~~ 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 account management object for this newly enabled event notification.

Parameters

`appAccountManager : in IpAppAccountManagerRef`

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

`chargingEventCriteria : in TpChargingEventCriteria`

Specifies the event specific criteria used by the application to define the charging event required. Individual addresses or address ranges may be specified for subscriber accounts. Example of events are "charging" and "recharging".

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_ADDRESS, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_UNKNOWN_SUBSCRIBER`

8.1.2 Method destroyNotification()

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

Parameters

`assignmentId : in TpAssignmentID`

Specifies the assignment ID that was given by the account management object when the application enabled the charging notification.

Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID`

8.1.3 Method queryBalanceReq()

This method is used by the application to query the balance of an account for one or several users.

Returns queryId : Specifies the ID of the balance query request.

Parameters

`users : in TpAddressSet`

Specifies the user(s) for which the balance is queried.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION`

8.1.4 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 manager interface for the event notification.

`eventCriteria : in TpChargingEventCriteria`

Specifies the new set of event 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, P_UNKNOWN_SUBSCRIBER, P_INVALID_ADDRESS`

8.1.5 Method getNotification()

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

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

Parameters

No Parameters were identified for this method

Returns

`TpChargingEventCriteriaResultSet`

Raises

`TpCommonExceptions`

8.1.6 Method retrieveTransactionHistoryReq()

This asynchronous method is used by the application to retrieve a transaction history of a subscriber's account. The history is a set of Detailed Records.

Returns retrievalID : Specifies the retrieval ID of the transaction history retrieval request.

Parameters

user : in TpAddress

Specifies the subscriber for whose account the transaction history is to be retrieved.

transactionInterval : in TpTimeInterval

Specifies the time interval for which the application history is to be retrieved.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION, P_INVALID_TIME_AND_DATE_FORMAT

8.1.7 Method <<new>> enableNotifications()

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

~~If the same application requests to enable notifications for a second time with a different IpAppAccountManager 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.~~

If the same application invokes this method multiple times with different IpAppAccountManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be 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

appAccountManager : in IpAppAccountManagerRef

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

8.1.8 Method <<new>> disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

End of Change in 8.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040655
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-12**	**CR**	**033**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0**	⌘

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

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

Title:	⌘	Support High Availability at API Level			
Source:	⌘	CN5 AePONA – Eamonn Murray			
Work item code:	⌘	OSA3	***Date:***	⌘	02/09/2004
Category:	⌘	**F**	***Release:***	⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Charging API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID, methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

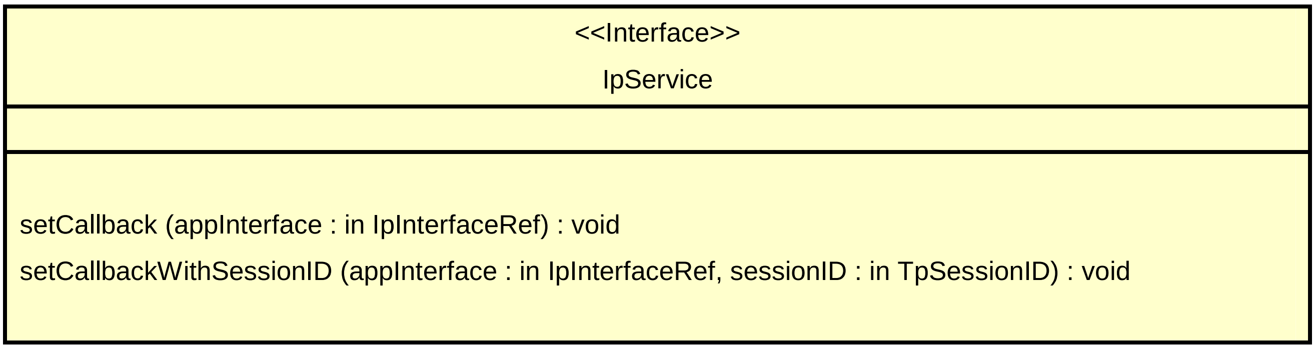
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.


<<Interface>>
IpService
setCallback (appInterface : in IpInterfaceRef) : void
setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

End of Change in 7.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040656**
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-13**	**CR**	**012**	⌘**rev**	**-**	⌘	Current version:	**6.2.0**	⌘

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

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

Title:	⌘	Support High Availability at API Level			
Source:	⌘	CN5 AePONA – Eamonn Murray			
Work item code:	⌘	OSA3	***Date:***	⌘	02/09/2004
Category:	⌘	**F**	***Release:***	⌘	*REL-6*

Use one of the following categories:
- ***F*** *(correction)*
- ***A*** *(corresponds to a correction in an earlier release)*
- ***B*** *(addition of feature),*
- ***C*** *(functional modification of feature)*
- ***D*** *(editorial modification)*

Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Policy Management API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID, methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

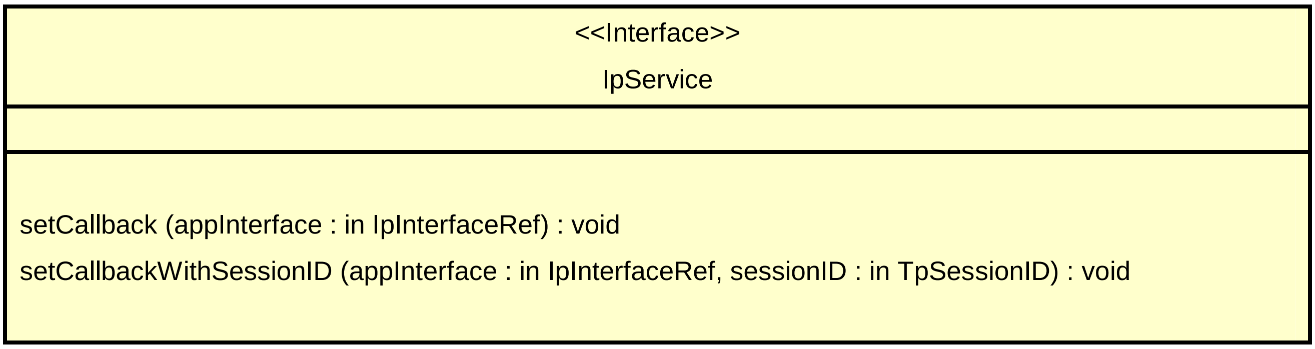
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface : in IpInterfaceRef) : void

setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE

End of Change in 7.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040657
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-14**	**CR**	**024**	⌘**rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Presence API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	The behaviour and operation of the setCallBack and setCallBackWithSessionID, methods have been clarified to indicate how mutliple callback references may be used.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	7.4.1

Other specs affected: ⌘

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

Other comments:	⌘	

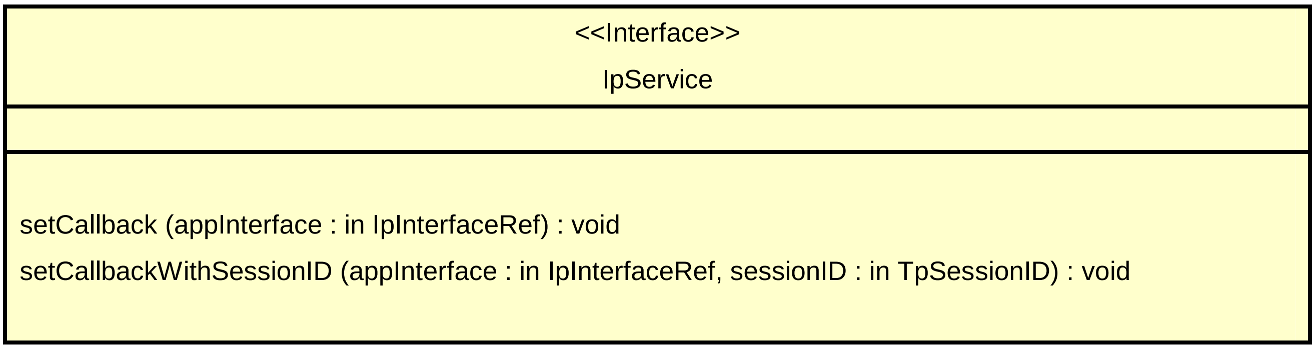
How to create CRs using this form:

Change in 7.4.1

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface : in IpInterfaceRef) : void

setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

End of Change in 7.4.1

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-040660
Meeting #28, Piscataway, New Jersey, USA, 10-14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-03**	**CR**	**126**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0**	⌘

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

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

Title:	⌘	Support High Availability at API Level		
Source:	⌘	CN5 AePONA – Eamonn Murray		
Work item code:	⌘	OSA3	***Date:*** ⌘	02/09/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Current application high availability that employs features of the OSA API is ambiguous and incomplete. Corrections and modifications are required to the Framework API in order to provide a complete specification that will support this feature in an unambigous and consistent fashion. These changes are submitted to fulfill the Release 6 stage 1 requirement for high availability for OSA at API level.
Summary of change:	⌘	Correct the framework API such that multiple identical application instances may establish an access session with the framework and thereafter obtain a reference to and use a common service manager. The behaviour and operation of several framework methods have been clarified to indicate their ability to support a deployment choice utilising multiple application instances. The changes are introduced in a manner that does not mandate the use of multiple application instances in order to achieve high availability, and vendors are free to implement an alternate high availability solution within their products.
Consequences if not approved:	⌘	The Release 6 stage 1 requirements cannot be addressed.

Clauses affected:	⌘	5.4.1, 6.3.1.2.1, 6.3.1.3, 7.1.4, 7.3.1, 7.3.2.2, 8.1.3				
		Y	N			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

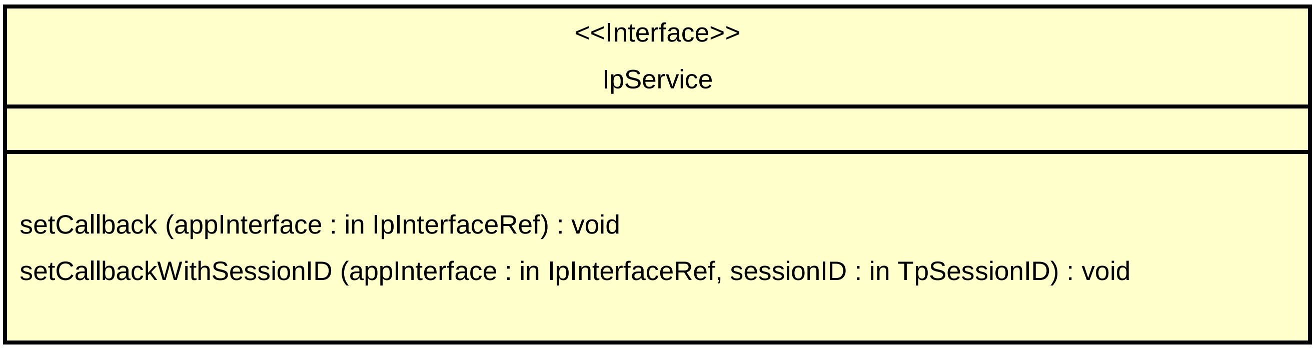
How to create CRs using this form:

Change in 5.4.1

5.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.


<<Interface>>
IpService
setCallback (appInterface : in IpInterfaceRef) : void
setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

5.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

Raises

TpCommonExceptions, P_INVALID_INTERFACE_TYPE

5.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks.

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE

End of Change in 5.4.1

Change in 6.3.1.2.1

6.3.1.2.1 Method terminateAccess()

The terminateAccess operation is used by the framework to end the client's access session.

After terminateAccess() is invoked, the client will no longer be authenticated with the framework. The client will not be able to use the references to any of the framework interfaces gained during the access session. Any calls to these interfaces will fail. ~~Also, all remaining service instances created by the framework either directly in this access session or on behalf of the client during this access session shall be terminated.~~ The framework shall also identify and terminate all remaining service instances that apply as a result of the client access termination. If at any point the framework's level of confidence in the identity of the client becomes too low, perhaps due to re-authentication failing, the framework should terminate all outstanding service agreements for that client, and should take steps to terminate the client's access session WITHOUT invoking terminateAccess() on the client. This follows a generally accepted security model where the framework has decided that it can no longer trust the client and will therefore sever ALL contact with it.

Parameters

terminationText : in TpString

This is the termination text describes the reason for the termination of the access session.

signingAlgorithm : in TpSigningAlgorithm

This is the algorithm used to compute the digital signature. It shall be identical to the one chosen by the framework in response to IpAccess.selectSigningAlgorithm(). If the signingAlgorithm is not the chosen one, is invalid, or unknown to the client, the P_INVALID_SIGNING_ALGORITHM exception will be thrown. The list of possible algorithms is as specified in the TpSigningAlgorithm table. The identifier used in this parameter must correspond to the digestAlgorithm and signatureAlgorithm fields in the SignerInfo field in the digitalSignature (see below).

digitalSignature : in TpOctetSet

This contains a CMS (Cryptographic Message Syntax) object (as defined in RFC 2630) with content type Signed-data. The signature is calculated and created as per section 5 of RFC 2630. The content is made of the termination text. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the termination text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention. The framework uses this to confirm its identity to the client. The client can check that the terminationText has been signed by the framework. If a match is made, the access session is terminated, otherwise the P_INVALID_SIGNATURE exception will be thrown.

Raises

TpCommonExceptions, P_INVALID_SIGNING_ALGORITHM, P_INVALID_SIGNATURE

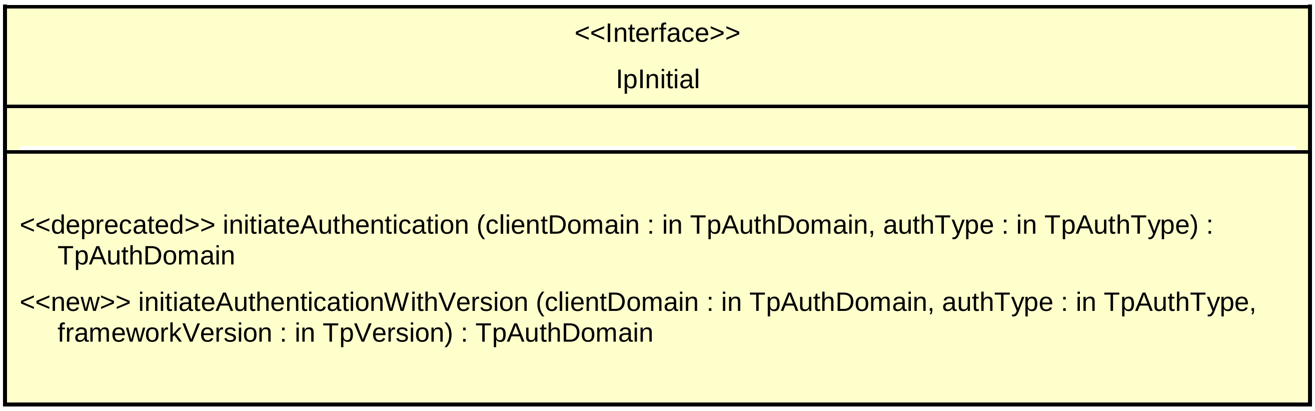
End of Change in 6.3.1.2.1

Change in 6.3.1.3

6.3.1.3 Interface Class IpInitial

Inherits from: IpInterface.

The Initial Framework interface is used by the client to initiate the authentication with the Framework. This interface shall be implemented by a Framework. The initiateAuthentication() and the initiateAuthenticationWithVersion() methods shall be implemented.



<<Interface>>

IpInitial

<<deprecated>> initiateAuthentication (clientDomain : in TpAuthDomain, authType : in TpAuthType) : TpAuthDomain

<<new>> initiateAuthenticationWithVersion (clientDomain : in TpAuthDomain, authType : in TpAuthType, frameworkVersion : in TpVersion) : TpAuthDomain

6.3.1.3.1 Method <<deprecated>> initiateAuthentication()

This method is deprecated in this version, this means that it will be supported until the next major release of the present document.

This method is invoked by the client to start the process of authentication with the framework, and request the use of a specific authentication method.

Returns <fwDomain> : This provides the client with a framework identifier, and a reference to call the authentication interface of the framework.

```
structure TpAuthDomain {
        domainID:         TpDomainID;
        authInterface:    IpInterfaceRef;
            };
```

The domainID parameter is an identifier for the framework (i.e. TpFwID). It is used to identify the framework to the client.

The authInterface parameter is a reference to the authentication interface of the framework. The type of this interface is defined by the authType parameter. The client uses this interface to authenticate with the framework.

Parameters

`clientDomain : in TpAuthDomain`

This identifies the client domain to the framework, and provides a reference to the ~~domain's~~ authentication interface.

```
structure TpAuthDomain {
        domainID:         TpDomainID;
        authInterface:    IpInterfaceRef;
            };
```

The domainID parameter is an identifier either for a client application (i.e. TpClientAppID) or for an enterprise operator (i.e. TpEntOpID), or for an instance of a service for which a client application has signed a service agreement (i.e. TpServiceInstanceID), or for a service supplier (i.e. TpServiceSupplierID). It is used to identify the client domain to the framework, (see authenticate() on IpAPILevelAuthentication). If the framework does not recognise the domainID, the framework returns an error code (P_INVALID_DOMAIN_ID).

A client application (identifiable by a given TpClientAppID) may optionally initiate authentication with the Framework by invoking this method multiple times. The Framework may elect to reject these subsequent requests, or may choose to associate them together as independent sessions under the same TpClientAppID.

The authInterface parameter is a reference to call the authentication interface of the client. The type of this interface is defined by the authType parameter. If the interface reference is not of the correct type, the framework returns an error code (P_INVALID_INTERFACE_TYPE).

`authType : in TpAuthType`

This identifies the type of authentication mechanism requested by the client. It provides operators and clients with the opportunity to use an alternative to the API level Authentication interface, e.g. an implementation specific authentication mechanism like CORBA Security, using the IpAuthentication interface, or Operator specific Authentication interfaces. OSA API level Authentication is the default authentication mechanism (P_OSA_AUTHENTICATION). If P_OSA_AUTHENTICATION is selected, then the clientDomain and fwDomain authInterface parameters are references to interfaces of type Ip(Client)APILevelAuthentication. If P_AUTHENTICATION is selected, the fwDomain authInterface parameter references to interfaces of type IpAuthentication which is used when an underlying distribution technology authentication mechanism is used.

Returns

`TpAuthDomain`

Raises

`TpCommonExceptions, P_INVALID_DOMAIN_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_AUTH_TYPE`

6.3.1.3.2 Method <<new>> initiateAuthenticationWithVersion()

This method is invoked by the client to start the process of authentication with the framework, and request the use of a specific authentication method using the new method with support for backward compatibility in the framework. The returned fwDomain authInterface will be selected to match the proposed version from the Client in the Framework response. If the Framework cannot work with the proposed framework version the framework returns an error code (P_INVALID_VERSION).

Returns <fwDomain> : This provides the client with a framework identifier, and a reference to call the authentication interface of the framework.

```
structure TpAuthDomain {
        domainID:       TpDomainID;
        authInterface:  IpInterfaceRef;
            };
```

The domainID parameter is an identifier for the framework (i.e. TpFwID). It is used to identify the framework to the client.

The authInterface parameter is a reference to the authentication interface of the framework that is unique for ~~the~~ each requesting client. The type of this interface is defined by the authType parameter. The client uses this interface to authenticate with the framework.

Note, there are no identifiers used in the authentication interface to correlate requests and responses, therefore the authentication interface may not be shared amongst multiple clients.

Parameters

`clientDomain : in TpAuthDomain`

This identifies the client domain to the framework, and provides a reference to the ~~domain's~~ authentication interface.

```
structure TpAuthDomain {
        domainID:        TpDomainID;
        authInterface:   IpInterfaceRef;
            };
```

The domainID parameter is an identifier either for a client application (i.e. TpClientAppID) or for an enterprise operator (i.e. TpEntOpID), or for an instance of a service for which a client application has signed a service agreement (i.e. TpServiceInstanceID), or for a service supplier (i.e. TpServiceSupplierID). It is used to identify the client domain to the framework, (see challenge() on IpAPILevelAuthentication). If the framework does not recognise the domainID, the framework returns an error code (P_INVALID_DOMAIN_ID).

A client application (identifiable by a given TpClientAppID) may optionally initiate authentication with the Framework by invoking this method multiple times. The Framework may elect to reject these subsequent requests, or may choose to associate them together as independent sessions under the same TpClientAppID.

The authInterface parameter is a reference to call the authentication interface of the client. The type of this interface is defined by the authType parameter. If the interface reference is not of the correct type, the framework returns an error code (P_INVALID_INTERFACE_TYPE).

`authType : in TpAuthType`

This identifies the type of authentication mechanism requested by the client. It provides operators and clients with the opportunity to use an alternative to the API level Authentication interface, e.g. an implementation specific authentication mechanism like CORBA Security, using the IpAuthentication interface, or Operator specific Authentication interfaces. OSA API level Authentication is the default authentication mechanism (P_OSA_AUTHENTICATION). If P_OSA_AUTHENTICATION is selected, then the clientDomain and fwDomain authInterface parameters are references to interfaces of type Ip(Client)APILevelAuthentication. If P_AUTHENTICATION is selected, the fwDomain authInterface parameter references to interfaces of type IpAuthentication that is used when an underlying distribution technology authentication mechanism is used.

`frameworkVersion : in TpVersion`

This identifies the version of the Framework implemented in the client. The TpVersion is a String containing the version number. Valid version numbers are defined in the respective framework specification.

Returns

`TpAuthDomain`

Raises

`TpCommonExceptions, P_INVALID_DOMAIN_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_AUTH_TYPE, P_INVALID_VERSION`

End of Change in 6.3.1.3

Change in 7.1.4

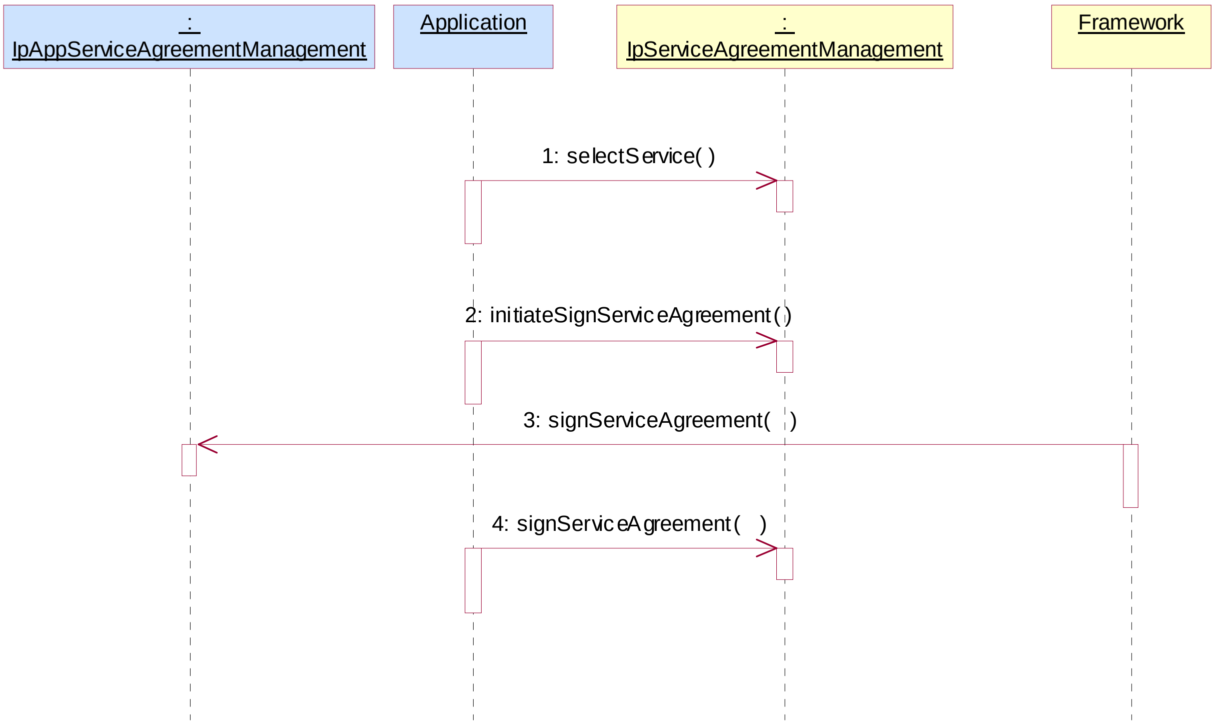
7.1.4 Service Agreement Management Sequence Diagrams

7.1.4.1 Service Selection

The following figure shows the process of selecting an SCF.

After discovery the Application gets a list of one or more SCF versions that match its required description. It now needs to decide which service it is going to use; it also needs to actually get a way to use it.

This is achieved by the following two steps:


: IpAppServiceAgreementManagement
Application
: IpServiceAgreementManagement
Framework
1: selectService( )
2: initiateSignServiceAgreement( )
3: signServiceAgreement( )
4: signServiceAgreement( )

1: Service Selection: first step - selectService

In this first step the Application identifies the SCF version it has finally decided to use. This is done by means of the serviceID, which is the agreed identifier for SCF versions. The Framework acknowledges this selection by returning to the Application ~~a new~~ an identifier for the service chosen: a service token, that is a private identifier for this service between this Application and this network, and is used for the process of signing the service agreement.

Input is:

- in serviceID

This identifies the SCF required.

And output:

- out serviceToken

This is a free format text token returned by the framework, which can be signed as part of a service agreement. It contains operator specific information relating to the service level agreement. An application (identifiable by a given TpClientAppID) may select the same service on more than one occasion in which case the same serviceToken, that identifies the relationship between the Application and the network, and the service agreement that applies, shall be returned. ~~Multiple selection of the same service may be used, for example, to allow the same application to be deployed multiple times in order to support a possible resilient application deployment configuration. Alternate approaches to achieving a resilient deployment, using a single service selection, shall also be possible.~~

2: Service Selection: second step - signServiceAgreement

In this second step an agreement is signed that allows the Application to use the chosen SCF version. And once this contractual details have been agreed, then the Application can be given the means to actually use it. The means are a reference to the manager interface of the SCF version (remember that a manager is an entry point to any SCF). By calling the createServiceManager operation on the lifecycle manager the Framework retrieves this interface and returns it to the Application. The service properties suitable for this application are also fed to the SCF (via the lifecycle manager interface) in order for the SCS to instantiate an SCF version that is suitable for this application.

The sequence of events indicated above, where the application initiates the signature process by calling initiateSignServiceAgreement, and where the framework calls signServiceAgreement on the application's IpAppServiceAgreementManagement interface before the application calls signServiceAgreement on the frameworks's IpServiceAgreementManagement, is the only sequence permitted.

Input:

- in serviceToken

This is the identifier that the network and Application have agreed to privately use for a certain version of SCF.

- in agreementText

This is the agreement text that is to be signed by the Framework using the private key of the Framework.

- in signingAlgorithm

This is the algorithm used to compute the digital signature.

Output:

- out signatureAndServiceMgr

This is a reference to a structure containing the digital signature of the Framework for the service agreement, and a reference to the manager interface of the SCF.

There must be only one service instance per client application. Therefore, in case an application (identifiable by a given TpClientAppID) attempts to select a service for which it has already signed a service agreement and this service agreement has not been terminated, the Framework may return a reference to the already existing service, or may raise an exception to the client indicating that this request is denied. ~~Multiple selection of the same service may be used, for example, to allow the same application to be deployed multiple times in order to support a possible resilient application deployment configuration. Alternate approaches to achieving a resilient deployment, using a single service selection, shall also be possible.~~

End of Change in 7.1.4

Change in 7.3.1

7.3.1 Service Discovery Interface Classes

7.3.1.1 Interface Class IpServiceDiscovery

Inherits from: IpInterface.

The service discovery interface, shown below, consists of four methods. Before a service can be discovered, the enterprise operator (or the client applications) must know what "types" of services are supported by the Framework and what service "properties" are applicable to each service type. The listServiceTypes() method returns a list of all "service types" that are currently supported by the framework and the "describeServiceType()" returns a description of each service type. The description of service type includes the "service-specific properties" that are applicable to each service type. Then the enterprise operator (or the client applications) can discover a specific set of registered services that both belong to a given type and possess the desired "property values", by using the "discoverService() method. Once the enterprise operator finds out the desired set of services supported by the framework, it subscribes to (a sub-set of) these services using the Subscription Interfaces. The enterprise operator (or the client applications in its domain) can find out the set of services available to it (i.e., the service that it can use) by invoking "listSubscribedServices()". The service discovery APIs are invoked by the enterprise operators or client applications. They are described below.

This interface shall be implemented by a Framework with as a minimum requirement the listServiceTypes(), describeServiceType() and discoverService() methods.

End of Change in 7.3.1

Change in 7.3.2.2

7.3.2.2 Interface Class IpServiceAgreementManagement

Inherits from: IpInterface.

This interface and the signServiceAgreement(), terminateServiceAgreement(), selectService() and initiateSignServiceAgreement() methods shall be implemented by a Framework.

<<Interface>>

IpServiceAgreementManagement

signServiceAgreement (serviceToken : in TpServiceToken, agreementText : in TpString, signingAlgorithm : in TpSigningAlgorithm) : TpSignatureAndServiceMgr

terminateServiceAgreement (serviceToken : in TpServiceToken, terminationText : in TpString, digitalSignature : in TpOctetSet) : void

selectService (serviceID : in TpServiceID) : TpServiceToken

initiateSignServiceAgreement (serviceToken : in TpServiceToken) : void

7.3.2.2.1 Method signServiceAgreement()

After the framework has called signServiceAgreement() on the application's IpAppServiceAgreementManagement interface, this method is used by the client application to request that the framework sign the service agreement, which allows the client application to use the service. A reference to the service manager interface of the service is returned to the client application. The service manager returned will be configured as per the service level agreement. If the framework uses service subscription, the service level agreement will be encapsulated in the subscription properties contained in the contract/profile for the client application, which will be a restriction of the registered properties. If the client application is not allowed to access the service, then an error code (P_SERVICE_ACCESS_DENIED) is returned. If the client application invokes this method before ~~the~~ processing (i.e. digital signature verification) the reponse of signServiceAgreement() on the application's IpAppServiceAgreementManagement interface completed, a TpCommonExceptions with ExceptionType P_INVALID_STATE may be raised to indicate that this method is currently unable to complete the method due to a race condition. In this case, the TpCommonExceptions with ExceptionType P_INVALID_STATE suggests the application to retry the method invocation after a reasonable amount of time has passed.

There must be only one service instance per client application. Therefore, in case the client attempts to select a service for which it has already signed a service agreement and this service agreement has not been terminated, a reference to the already existing service manager will be returned. ~~Multiple selection of the same service may be used, for example, to allow the same application to be deployed multiple times in order to support a possible resilient application deployment configuration. Alternate approaches to achieving a resilient deployment, using a single service selection, shall also be possible.~~

Returns <signatureAndServiceMgr> : This contains the digital signature of the framework for the service agreement, and a reference to the service manager interface of the service.

```
structure TpSignatureAndServiceMgr {
        digitalSignature:  TpOctetSet;
          serviceMgrInterface:  IpServiceRef;
            };
```

The digitalSignature contains a CMS (Cryptographic Message Syntax) object (as defined in RFC 2630) with content type Signed-data. The signature is calculated and created as per section 5 of RFC 2630. The content is the agreement text given by the client application. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the agreement text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention.

The serviceMgrInterface is a reference to the service manager interface for the selected service.

Parameters

serviceToken : in TpServiceToken

This is the token returned by the framework in a call to the selectService() method. This token is used to identify the service instance requested by the client application. If the serviceToken is invalid, or has expired, an error code (P_INVALID_SERVICE_TOKEN) is returned.

agreementText : in TpString

This is the agreement text that is to be signed by the framework using the private key of the framework. If the agreementText is invalid, then an error code (P_INVALID_AGREEMENT_TEXT) is returned.

signingAlgorithm : in TpSigningAlgorithm

This is the algorithm used to compute the digital signature. It shall be identical to the one chosen by the framework in response to IpAccess.selectSigningAlgorithm(). If the signingAlgorithm is not the chosen one, is invalid, or unknown to the framework, an error code (P_INVALID_SIGNING_ALGORITHM) is returned. The list of possible algorithms is as specified in the TpSigningAlgorithm table. The identifier used in this parameter must correspond to the digestAlgorithm and signatureAlgorithm fields in the SignerInfo field in the digitalSignature (see below).

Returns

`TpSignatureAndServiceMgr`

Raises

`TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_AGREEMENT_TEXT, P_INVALID_SERVICE_TOKEN, P_INVALID_SIGNING_ALGORITHM, P_SERVICE_ACCESS_DENIED`

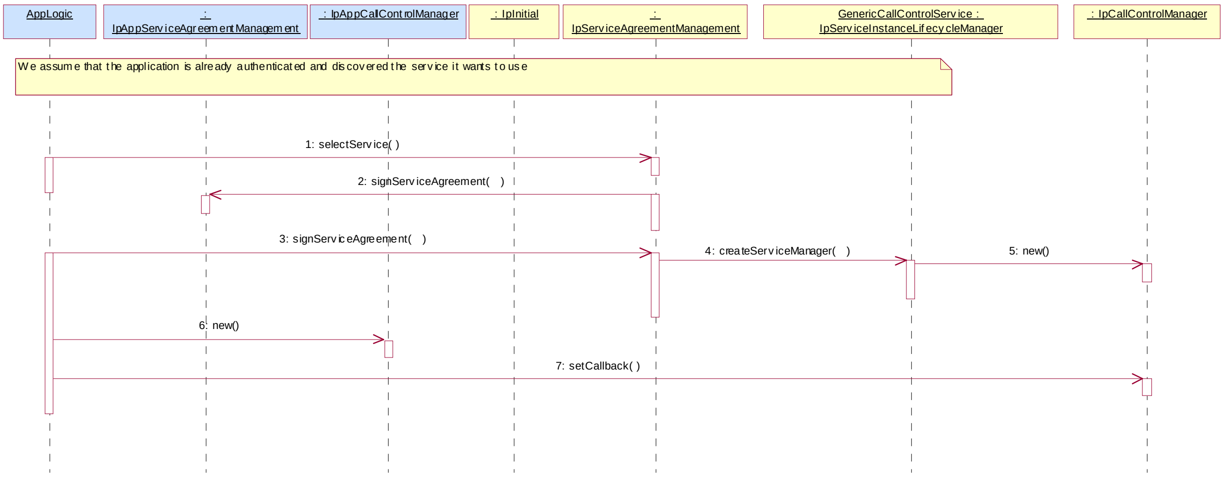
End of Change in 7.3.2.2

Change in 8.1.3

8.1.3 Service Instance Lifecycle Manager Sequence Diagrams

8.1.3.1 Sign Service Agreement

This sequence illustrates how the application can get access to a specified service. It only illustrates the last part: the signing of the service agreement and the corresponding actions towards the service. For more information on accessing the framework, authentication and discovery of services, see the corresponding clauses.


AppLogic
:
IpAppServiceAgreementManagement
: IpAppCallControlManager
: IpInitial
:
IpServiceAgreementManagement
GenericCallControlService :
IpServiceInstanceLifecycleManager
: IpCallControlManager
We assume that the application is already authenticated and discovered the service it wants to use
1: selectService( )
2: signServiceAgreement( )
3: signServiceAgreement( )
4: createServiceManager( )
5: new()
6: new()
7: setCallback( )

1: The application selects the service, using a serviceID for the generic call control service. The serviceID could have been obtained via the discovery interface. A ServiceToken is returned to the application.

2: The client application signs the service agreement.

3: The framework signs the service agreement. As a result a service manager interface reference (in this case of type IpCallControlManager) is returned to the application.

4: Provided the signature information is correct and all conditions have been fulfilled, the framework will request the service identified by the serviceID to return a service manager interface reference. The service manager is the initial point of contact to the service.

5: The lifecycle manager creates a new manager interface instance (a call control manager) for the specified application. It should be noted that this is an implementation detail. The service implementation may use other mechanism to get a service manager interface instance.

Following the creation of the service manager outlined above, a unique instance of the service particular to the application client results. This service instance is assigned a serviceInstanceID by the Framework, which is provided to the Service Instance Lifecycle manager during the createServiceManager operation. If it is necessary that Framework Integrity Management functionality and operations are to be supported between the Framework and the service instance identified by the defined serviceInstanceID, it is then necessary for the new service instance to establish an access session with the Framework. This provides the Framework with the ability to manage and monitor the operation of the service instance that relates to a particular application client. The steps required to establish a Framework access session are outlined in chapter 6 of this specification.

6: The application creates a new IpAppCallControlManager interface to be used for callbacks.

7: The Application sets the callback interface to the interface created with the previous message.

An application (identifiable by a given TpClientAppID may carry out the sequence, as exemplified above, multiple times. ~~This approach may be used, for example, to allow the same application to be deployed multiple times in order to support a possible resilient application deployment configuration. Alternate approaches to achieving a resilient deployment, using a single signServiceAgreement sequence shall also be possible.~~

End of Change in 8.1.3