3GPP TSG CN Plenary Meeting #16 **NP-020189**
5th - 7th June 2002. Marco Island, USA.

Source:	**CN5 (OSA)**
Title:	**Rel-5 CRs 29.198-05 OSA API Part 5: Generic user interaction**
Agenda item:	**8.2**
Document for:	**APPROVAL**

Doc-1st -Level	Spec	CR	Rv	Pha	Subject	Cat	Ver Curr	Ver New	Doc-2nd -Level	Work item
NP-020189	29.198-05	010	-	Rel-5	Improve the vague description of P_ID_NOT_FOUND	D	4.4.0	5.0.0	N5-020377	OSA2
NP-020189	29.198-05	012	-	Rel-5	Detach call leg before playing announcement or collecting digits	F	4.4.0	5.0.0	N5-020475	OSA2
NP-020189	29.198-05	013	-	Rel-5	Delete P_INVALID_CRITERIA from sendInfoAndCollectReq()	F	4.4.0	5.0.0	N5-020481	OSA2
NP-020189	29.198-05	015	-	Rel-5	Correcting erroneous description of UI behaviour in call control	F	4.4.0	5.0.0	N5-020501	OSA2

joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020377**
Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002

CR-Form-v5

CHANGE REQUEST

⌘	**29.198-05**	**CR**	**010**	⌘ **rev**	**-**	⌘	Current version:	**4.4.0** ⌘

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

Proposed change affects: ⌘	(U)SIM	☐	ME/UE	☐	Radio Access Network	☐	Core Network	X

Title:	⌘	Improve the vague description of P_ID_NOT_FOUND		
Source:	⌘	CN5		
Work item code:	⌘	OSA2	***Date:*** ⌘	17/05/2002
Category:	⌘	**D**	***Release:*** ⌘	*REL-5*
		Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.	*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*	

Reason for change:	⌘	Vague descriptions of P_ID_NOT_FOUND may lead to confusion.
Summary of change:	⌘	The description of P_ID_NOT_FOUND in this Part 5 (29.198-05) is improved.
Consequences if not approved:	⌘	

Clauses affected:	⌘	11		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘			

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

11 Exception Classes

The following are the list of exception classes which are used in this interface of the API.

Name	Description
P_ILLEGAL_ID	Information id specified is invalid
P_ID_NOT_FOUND	~~A legal i~~Information id is ~~not~~ unknown ~~to the User Interaction Service~~
P_ILLEGAL_RANGE	The values for minimum and maximum collection length are out of range.
P_INVALID_COLLECTION_CRITERIA	Invalid collection criteria specified

joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020475**
Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002

CR-Form-v5

CHANGE REQUEST

⌘	**29.198-05**	**CR**	**012**	⌘ **rev**	**-**	⌘	Current version:	**4.4.0**	⌘

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

Proposed change affects: ⌘	(U)SIM	☐	ME/UE	☐	Radio Access Network	☐	Core Network	X

Title:	⌘	Detach call leg before playing announcement or collecting digits		
Source:	⌘	CN5		
Work item code:	⌘	OSA2	***Date:*** ⌘	17/05/2002
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-5* *Use one of the following releases:* 2 *(GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change:	⌘	In case the application wants to play an announcement to or get digits from one one party in the call, it can add one callLeg to a UICall object. However, it is not clear from the spec that the CallLeg should be detached from the Call so that the party corresponding to the specific CallLeg is the only one to get the announcement.
Summary of change:	⌘	Add clarification to indicate that a CallLeg should be detached before user interaction can be accepted.
Consequences if not approved:	⌘	Interworking problems.

Clauses affected:	⌘	11.22		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘			

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

11.22 TpUITargetObjectType

Defines the type of object where User Interaction should be performed upon.

Name	Value	Description
P_UI_TARGET_OBJECT_CALL	0	User-interaction will be performed on a complete Call.
P_UI_TARGET_OBJECT_MULTI_PARTY_CALL	1	User-interaction will be performed on a complete Multi-party Call.
P_UI_TARGET_OBJECT_CALL_LEG	2	User-interaction will be performed on a single Call Leg. The media of this call leg should be detached at the moment any user interaction is done.

joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020481**
Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002

CR-Form-v5

CHANGE REQUEST

⌘ **29.198-05 CR 013** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

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

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network [✓ X]

Title: ⌘	Delete P_INVALID_CRITERIA from sendInfoAndCollectReq()		
Source: ⌘	CN5		
Work item code: ⌘	OSA2	***Date:*** ⌘	17/05/2002
Category: ⌘	**F**	***Release:*** ⌘	*REL-5*
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change: ⌘	P_INVALID_CRITERIA and P_INVALID_COLLECTION_CRITERIA are identical and have identical uses in sendInfoAndCollectReq().
Summary of change: ⌘	P_INVALID_CRITERIA in this Part 5 is deleted.
Consequences if not approved: ⌘	Conflicting use of P_INVALID_CRITERIA and P_INVALID_COLLECTION_CRITERIA may lead to confusion.

Clauses affected: ⌘	8.3	
Other specs affected: ⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘
Other comments: ⌘		

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

8 Generic User Interaction Interface Classes

The Generic User Interaction Service interface (GUIS) is used by applications to interact with end users. The GUIS is represented by the IpUIManager, IpUI and IpUICall interfaces that interface to services provided by the network. To handle responses and reports, the developer must implement IpAppUIManager and IpAppUI interfaces to provide the callback mechanism.

8.1 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.

<<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

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`

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`

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

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`

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`

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`

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`

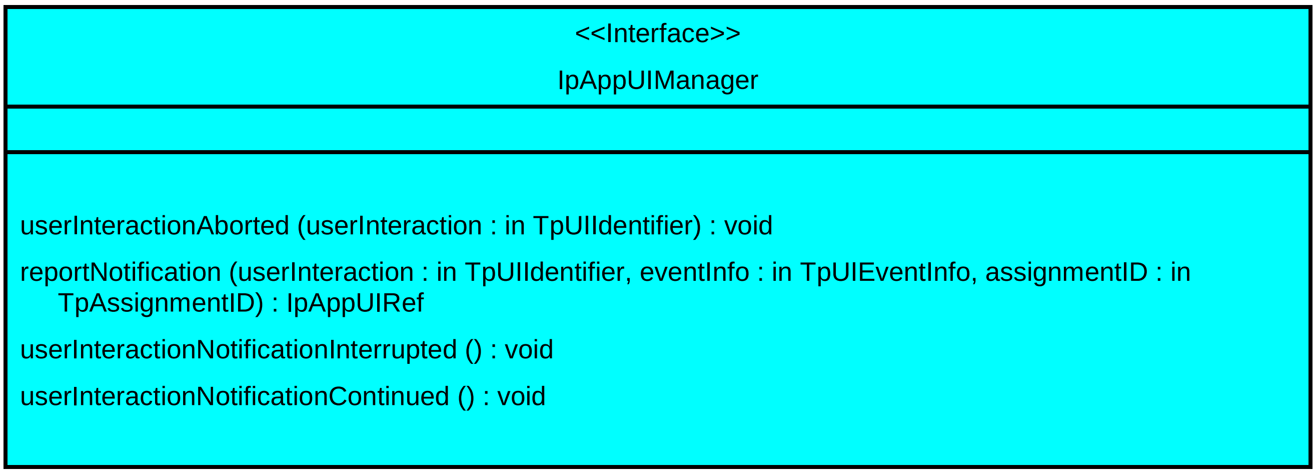
Raises

`TpCommonExceptions, P_INVALID_CRITERIA`

8.2 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
reportNotification (userInteraction : in TpUIIdentifier, eventInfo : in TpUIEventInfo, assignmentID : in TpAssignmentID) : IpAppUIRef
userInteractionNotificationInterrupted () : void
userInteractionNotificationContinued () : void

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.

Method

`reportNotification()`

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.

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`

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

Method

`userInteractionNotificationContinued()`

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

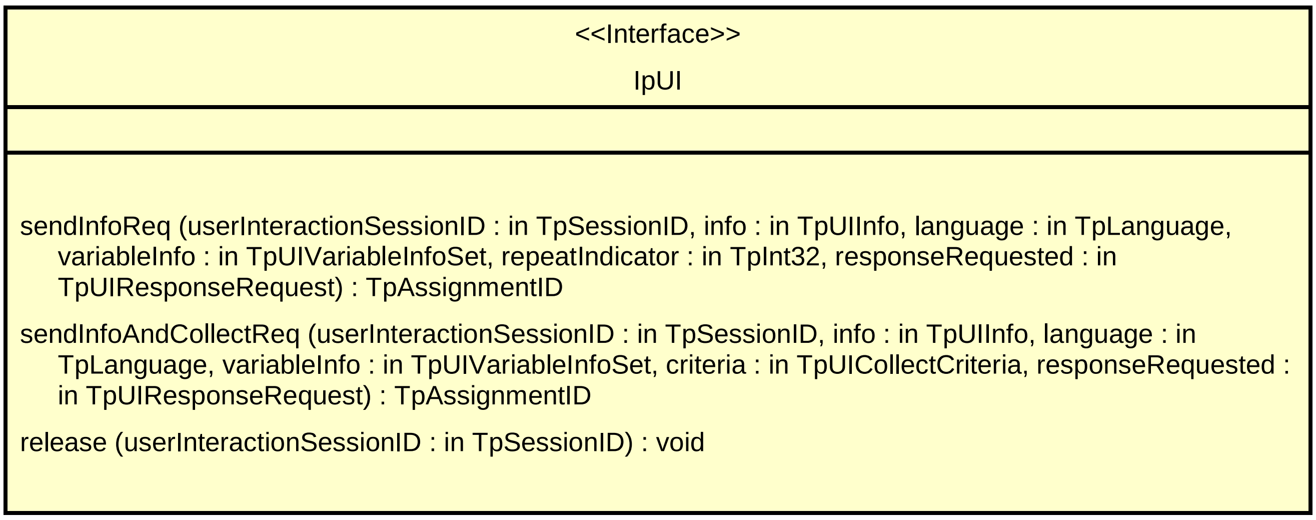
Parameters

No Parameters were identified for this method

8.3 Interface Class IpUI

Inherits from: IpService.

The User Interaction Service Interface provides functions to send information to, or gather information from the user. An application can use the User Interaction Service Interface independently of other services.



<<Interface>>

IpUI

sendInfoReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, language : in TpLanguage, variableInfo : in TpUIVariableInfoSet, repeatIndicator : in TpInt32, responseRequested : in TpUIResponseRequest) : TpAssignmentID

sendInfoAndCollectReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, language : in TpLanguage, variableInfo : in TpUIVariableInfoSet, criteria : in TpUICollectCriteria, responseRequested : in TpUIResponseRequest) : TpAssignmentID

release (userInteractionSessionID : in TpSessionID) : void

Method

sendInfoReq()

This asynchronous method plays an announcement or sends other information to the user.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

info : in TpUIInfo

Specifies the information to send to the user. This information can be:

- an infoID, identifying pre-defined information to be send (announcement and/or text);

- a string, defining the text to be sent;

- a URL , identifying pre-defined information or data to be sent to or downloaded into the terminal.

language : in TpLanguage

Specifies the Language of the information to be send to the user.

variableInfo : in TpUIVariableInfoSet

Defines the variable part of the information to send to the user.

repeatIndicator : in TpInt32

Defines how many times the information shall be sent to the end-user. A value of zero (0) indicates that the announcement shall be repeated until the call or call leg is released or an abortActionReq() is sent.

responseRequested : in TpUIResponseRequest

Specifies if a response is required from the call user interaction service, and any action the service should take.

Returns

TpAssignmentID

Raises

TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE,P_ILLEGAL_ID,P_ID_NOT_FOUND

Method

sendInfoAndCollectReq()

This asynchronous method plays an announcement or sends other information to the user and collects some information from the user. The announcement usually prompts for a number of characters (for example, these are digits or text strings such as "YES" if the user's terminal device is a phone).

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

info : in TpUIInfo

Specifies the ID of the information to send to the user. This information can be:

- an infoID, identifying pre-defined information to be send (announcement and/or text);

- a string, defining the text to be sent;

- a URL , identifying pre-defined information or data to be sent to or downloaded into the terminal

language : in TpLanguage

Specifies the Language of the information to be send to the user.

variableInfo : in TpUIVariableInfoSet

Defines the variable part of the information to send to the user.

criteria : in TpUICollectCriteria

Specifies additional properties for the collection of information, such as the maximum and minimum number of characters, end character, first character timeout and inter-character timeout.

responseRequested : in TpUIResponseRequest

Specifies if a response is required from the call user interaction service, and any action the service should take. For this case it can especially be used to indicate e.g. the final request.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE,P_ILLEGAL_ID,P_ID_NOT_FOUND`~~`,P_INVALID_CRITERIA`~~`,P_ILLEGAL_RANGE,P_INVALID_COLLECTION_CRITERIA`

Method
`release()`

This method requests that the relationship between the application and the user interaction object be released. It causes the release of the used user interaction resources and interrupts any ongoing user interaction.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction created.

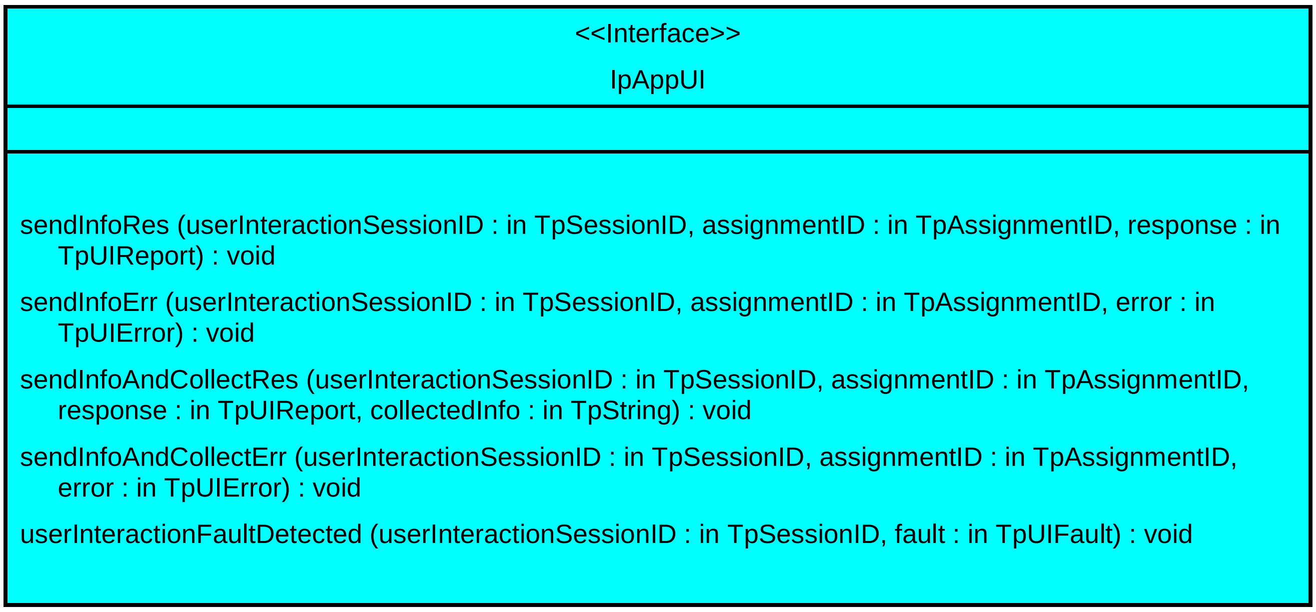
Raises

`TpCommonExceptions,P_INVALID_SESSION_ID`

8.4 Interface Class IpAppUI

Inherits from: IpInterface.

The User Interaction Application Interface is implemented by the client application developer and is used to handle generic user interaction request responses and reports.



<<Interface>>

IpAppUI

sendInfoRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport) : void

sendInfoErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

sendInfoAndCollectRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport, collectedInfo : in TpString) : void

sendInfoAndCollectErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

userInteractionFaultDetected (userInteractionSessionID : in TpSessionID, fault : in TpUIFault) : void

Method
sendInfoRes()

This asynchronous method informs the application about the completion of a sendInfoReq(). This response is called only if the responseRequested parameter of the sendInfoReq() method was set to P_UICALL_RESPONSE_REQUIRED.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

response : in TpUIReport

Specifies the type of response received from the user.

Method
sendInfoErr()

This asynchronous method indicates that the request to send information was unsuccessful.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

error : in TpUIError

Specifies the error which led to the original request failing.

Method
sendInfoAndCollectRes()

This asynchronous method returns the information collected to the application.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

response : in TpUIReport

Specifies the type of response received from the user.

`collectedInfo : in TpString`

Specifies the information collected from the user.

Method

`sendInfoAndCollectErr()`

This asynchronous method indicates that the request to send information and collect a response was unsuccessful.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

`error : in TpUIError`

Specifies the error which led to the original request failing.

Method

`userInteractionFaultDetected()`

This method indicates to the application that a fault has been detected in the user interaction.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the interface and sessionID of the user interaction service in which the fault has been detected.

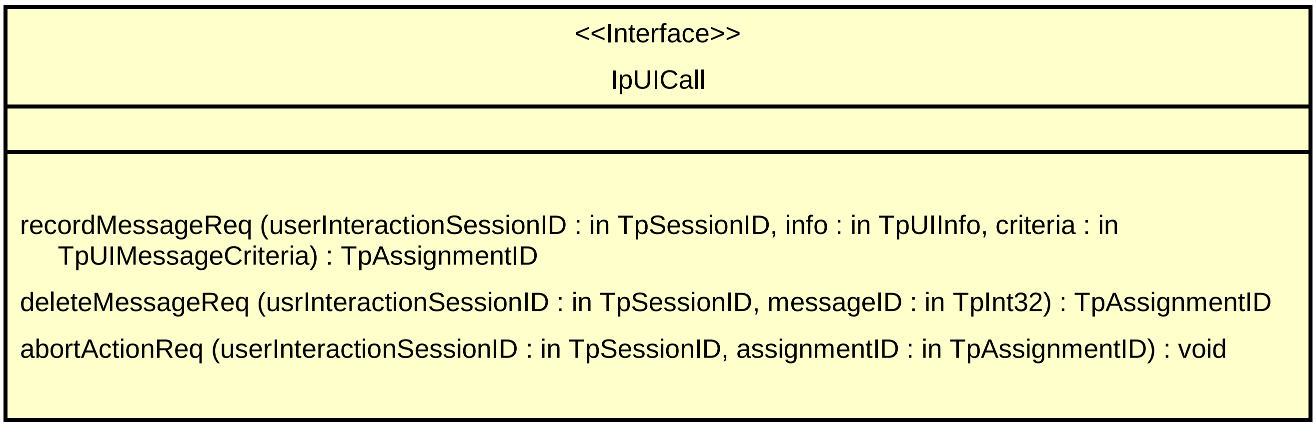
`fault : in TpUIFault`

Specifies the fault that has been detected.

8.5 Interface Class IpUICall

Inherits from: IpUI.

The Call User Interaction Service Interface provides functions to send information to, or gather information from the user (or call party) to which a call leg is connected. An application can use the Call User Interaction Service Interface only in conjunction with another service interface, which provides mechanisms to connect a call leg to a user. At present, only the Call Control service supports this capability.



<<Interface>>

IpUICall

recordMessageReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, criteria : in TpUIMessageCriteria) : TpAssignmentID

deleteMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID

abortActionReq (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void

Method
`recordMessageReq()`

This asynchronous method allows the recording of a message. The recorded message can be played back at a later time with the sendInfoReq() method.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`info : in TpUIInfo`

Specifies the information to send to the user. This information can be either an ID (for pre-defined announcement or text), a text string, or an URL (indicating the information to be sent, e.g. an audio stream).

`criteria : in TpUIMessageCriteria`

Defines the criteria for recording of messages

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE,P_ILLEGAL_ID,P_ID_NOT_FOUND,P_INVALID_CRITERIA`

Method
`deleteMessageReq()`

This asynchronous method allows to delete a recorded message.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`messageID : in TpInt32`

Specifies the message ID.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_ILLEGAL_ID,P_ID_NOT_FOUND`

Method

`abortActionReq()`

This asynchronous method aborts a user interaction operation, e.g. a sendInfoReq(), from the specified call leg. The call and call leg are otherwise unaffected. The user interaction call service interrupts the current action on the specified leg.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the user interaction request to be cancelled.

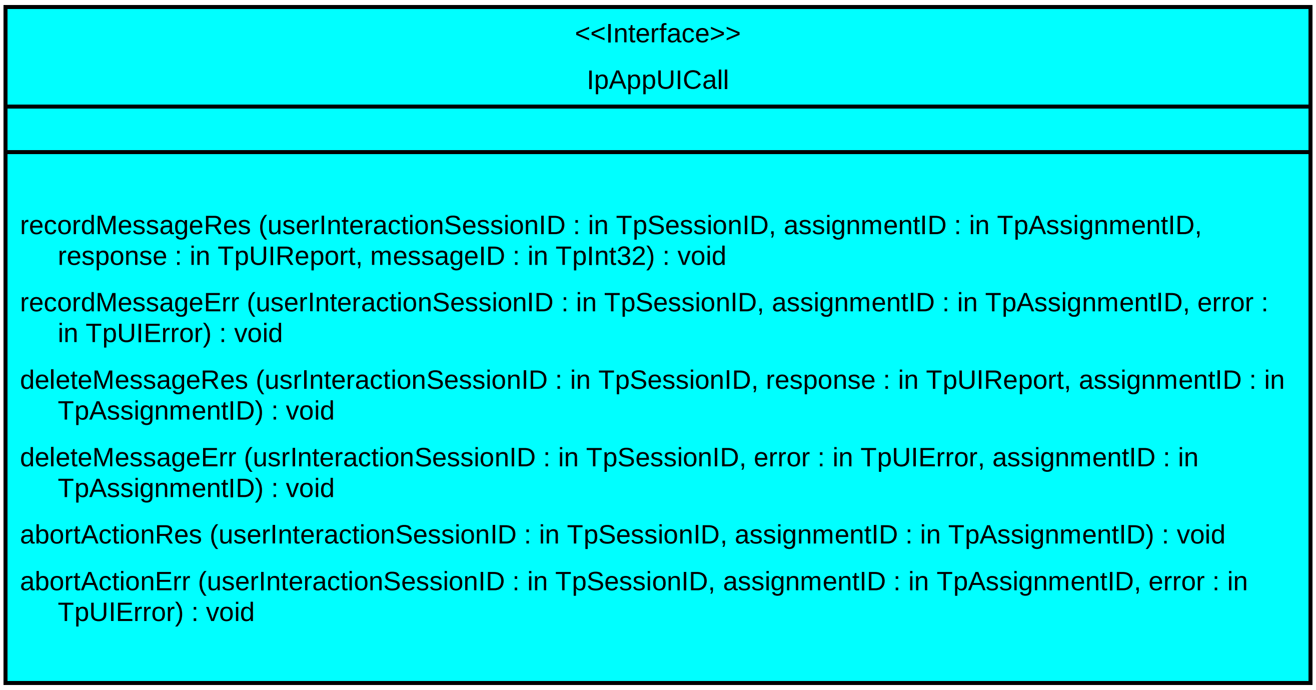
Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_ASSIGNMENT_ID`

8.6 Interface Class IpAppUICall

Inherits from: IpAppUI.

The Call User Interaction Application Interface is implemented by the client application developer and is used to handle call user interaction request responses and reports.



<<Interface>>

IpAppUICall

recordMessageRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport, messageID : in TpInt32) : void

recordMessageErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

deleteMessageRes (usrInteractionSessionID : in TpSessionID, response : in TpUIReport, assignmentID : in TpAssignmentID) : void

deleteMessageErr (usrInteractionSessionID : in TpSessionID, error : in TpUIError, assignmentID : in TpAssignmentID) : void

abortActionRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void

abortActionErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

Method
recordMessageRes()

This method returns whether the message is successfully recorded or not. In case the message is recorded, the ID of the message is returned.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the call user interaction interface for a user interaction request.

response : in TpUIReport

Specifies the type of response received from the device where the message is stored.

messageID : in TpInt32

Specifies the ID that was assigned to the message by the device where the message is stored.

Method
recordMessageErr()

This method indicates that the request for recording of a message was not successful.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the call user interaction interface for a user interaction request.

error : in TpUIError

Specifies the error which led to the original request failing.

Method
deleteMessageRes()

This method returns whether the message is successfully deleted or not.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

response : in TpUIReport

Specifies the type of response received from the device where the message was stored.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the call user interaction interface for a user interaction request.

Method
deleteMessageErr()

This method indicates that the request for deleting a message was not successful.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

error : in TpUIError

Specifies the error which led to the original request failing.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the call user interaction interface for a user interaction request.

Method
abortActionRes()

This asynchronous method confirms that the request to abort a user interaction operation on a call leg was successful.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the call user interaction interface for a user interaction request.

Method
`abortActionErr()`

This asynchronous method indicates that the request to abort a user interaction operation on a call leg resulted in an error.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the call user interaction interface for a user interaction request.

`error : in TpUIError`

Specifies the error which led to the original request failing.

joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020501**
Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002

CR-Form-v5

CHANGE REQUEST

⌘	**29.198-05**	**CR**	**015**	⌘ **rev**	**-**	⌘	Current version:	**4.4.0** ⌘

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

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network [✓ X]

Title: ⌘	Correcting erroneous description of UI behaviour in call control		
Source: ⌘	CN5		
Work item code: ⌘	OSA2	***Date:*** ⌘	30/05/2002
Category: ⌘	**F**	***Release:*** ⌘	*REL-5*
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change: ⌘	5.2 reference the framework but should reference the call control service. 5.3 incorrectly describes the behaviour of UI in Generic Call Control, as it states that it is possible in GCC to play announcements to a specific party. This is incorrect as there is no access to call legs in GCC.
Summary of change: ⌘	Change 5.3 so that it more accurately describes the functionality being illustrated in the diagram.
Consequences if not approved: ⌘	The description of the diagram will conflict with the actual functionality offered in GCC, leading to incorrect implementations and interoperability problems.

Clauses affected: ⌘	5.2, 5.3	
Other specs affected: ⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications	
Other comments: ⌘		

How to create CRs using this form:

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

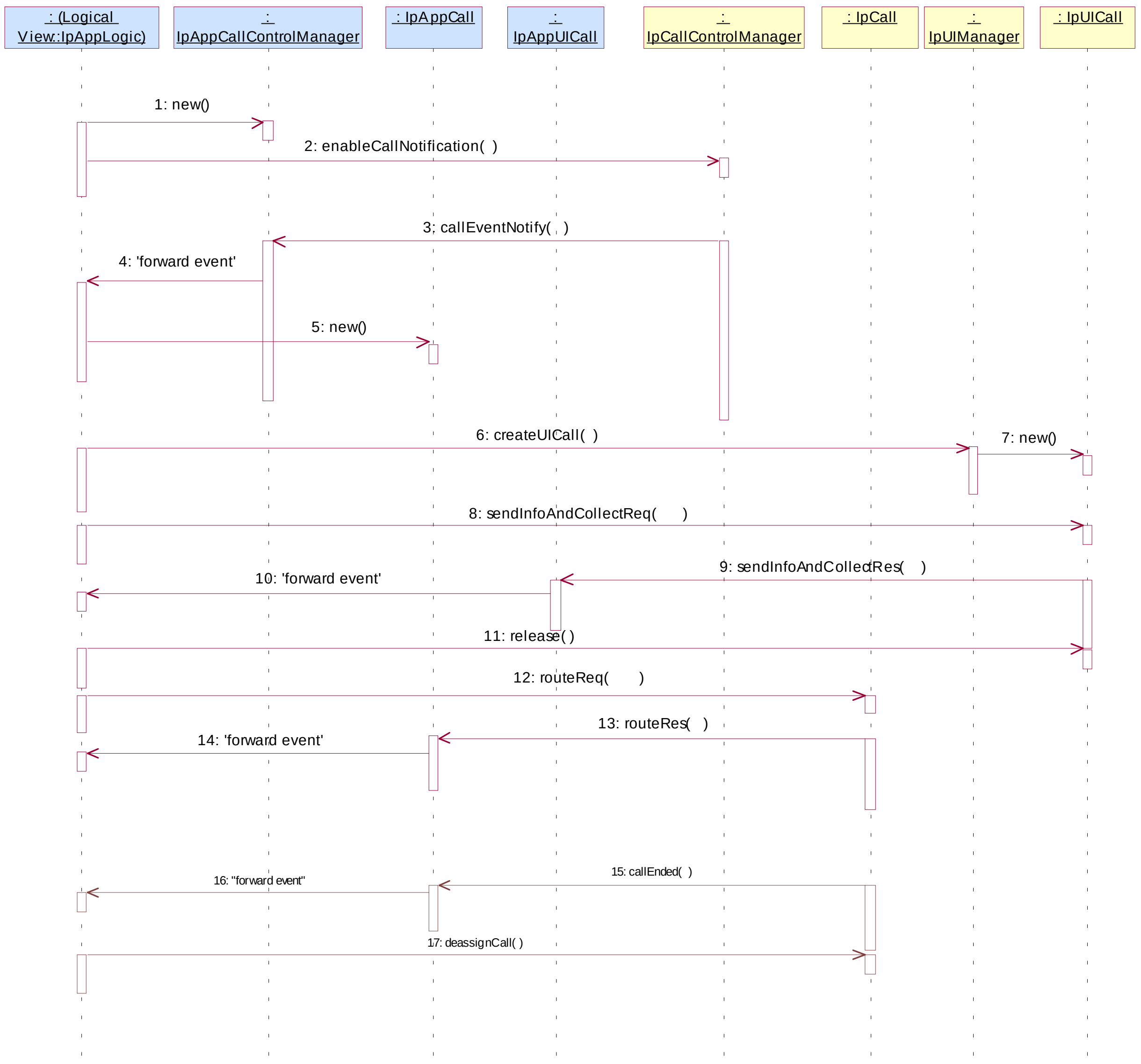
1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

5.2 Call Barring 1

The following sequence diagram shows a call barring service, initiated as a result of a prearranged event being received by the ~~framework~~call control service. Before the call is routed to the destination number, the calling party is asked for a PIN code. The code is accepted and the call is routed to the original called party.



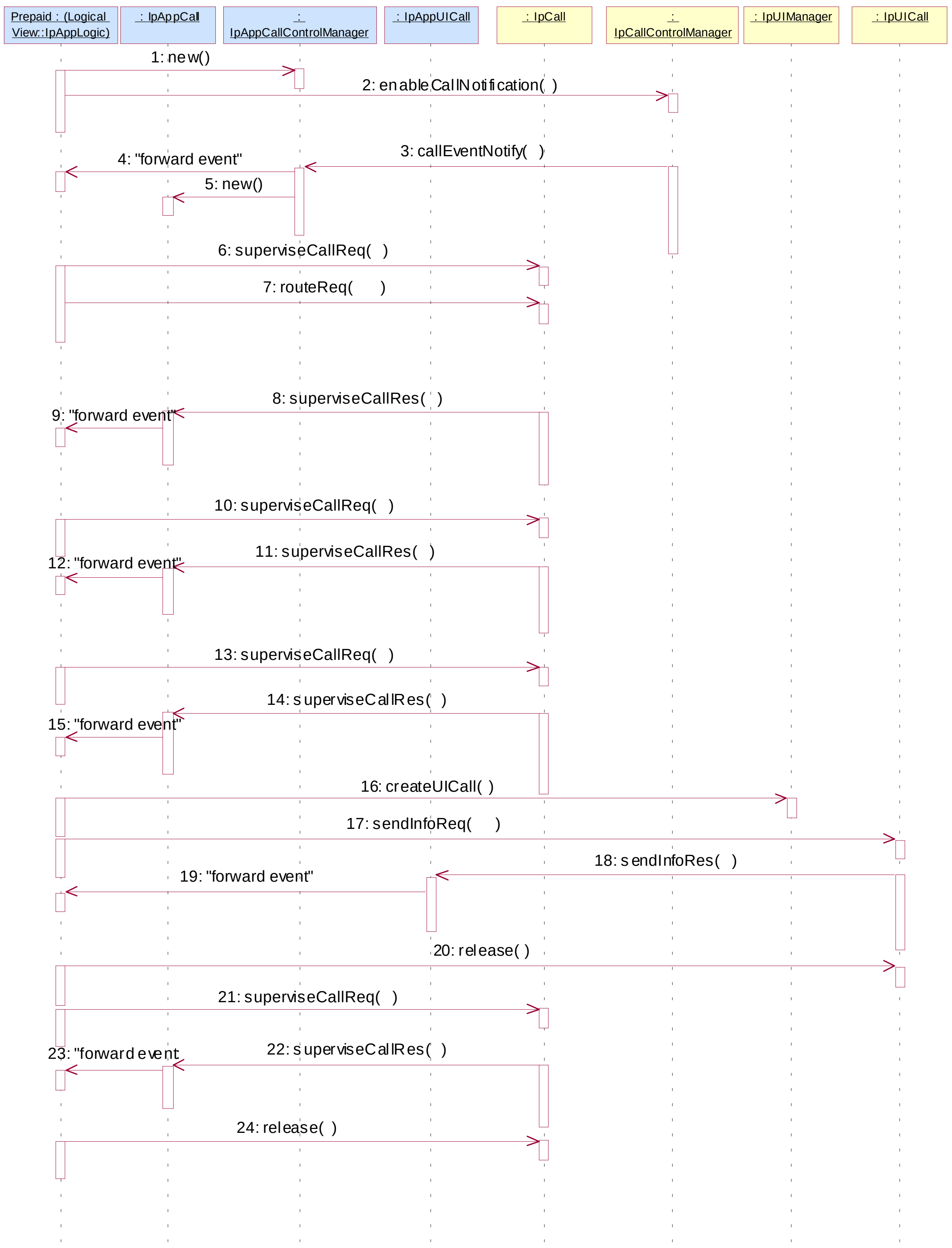
1: This message is used by the application to create an object implementing the IpAppCallControlManager interface.

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range prompted for a password before the call is allowed to progress. When a new call, that matches the event criteria set, arrives, a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.
4: This message is used to forward the previous message to the IpAppLogic.
5: This message is used by the application to create an object implementing the IpAppCall interface. The reference to this object is passed back to the object implementing the IpCallControlManager using the return parameter of the callEventNotify.
6: This message is used to create a new UICall object. The reference to the call object is given when creating the UICall.
7: Provided all the criteria are fulfilled, a new UICall object is created.
8: The call barring service dialogue is invoked.
9: The result of the dialogue, which in this case is the PIN code, is returned to its callback object.
10: This message is used to forward the previous message to the IpAppLogic.
11: This message releases the UICall object.
12: Assuming the correct PIN is entered, the call is forward routed to the destination party.
13: This message passes the result of the call being answered to its callback object.
14: This message is used to forward the previous message to the IpAppLogic
15: When the call is terminated in the network, the application will receive a notification. This notification will always be received when the call is terminated by the network in a normal way, the application does not have to request this event explicitly.
16: The event is forwarded to the application.
17: The application must free the call related resources in the gateway by calling deassignCall.

5.3 Prepaid

This sequence shows a Pre-paid application. The subscriber is using a pre-paid card or credit card to pay for the call. The application each time allows a certain timeslice for the call. After the timeslice, a new timeslice can be started or the application can terminate the call. In the following sequence the end-user will received an announcement before his final timeslice.



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

2:	This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met, other messages (not shown) are used to create the call and associated call leg object.
3:	The incoming call triggers the Pre-Paid Application (PPA).
4:	The message is forwarded to the application.
5:	A new object on the application side for the Generic Call object is created
6:	The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.
7:	Before continuation of the call, PPA sends all charging information, a possible tariff switch time and the call duration supervision period, towards the GW which forwards it to the network.
8:	At the end of each supervision period the application is informed and a new period is started.
9:	The message is forwarded to the application.
10:	The Pre-Paid Application (PPA) requests to supervise the call for another call duration.
11:	At the end of each supervision period the application is informed and a new period is started.
12:	The message is forwarded to the application.
13:	The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.
14:	When the user is almost out of credit the application is informed.~~an announcement is played to inform about this. The announcement is played only to the leg of the A-party, the B-party will not hear the announcement.~~
15:	The message is forwarded to the application.
16:	The application decides to play an announcement to the parties in this call. A new UICall object is created and associated with the ~~controlling leg~~call.
17:	An announcement is played ~~to the controlling leg~~ informing the user about the near-expiration of his credit limit. The B-subscriber will not hear the announcement.
18:	When the announcement is completed the application is informed.
19:	The message is forwarded to the application.
20:	The application releases the UICall object.
21:	The user does not terminate so the application terminates the call after the next supervision period.
22:	The supervision period ends
23:	The event is forwarded to the logic.
24:	The application terminates the call. Since the user interaction is already explicitly terminated no userInteractionFaultDetected is sent to the application.