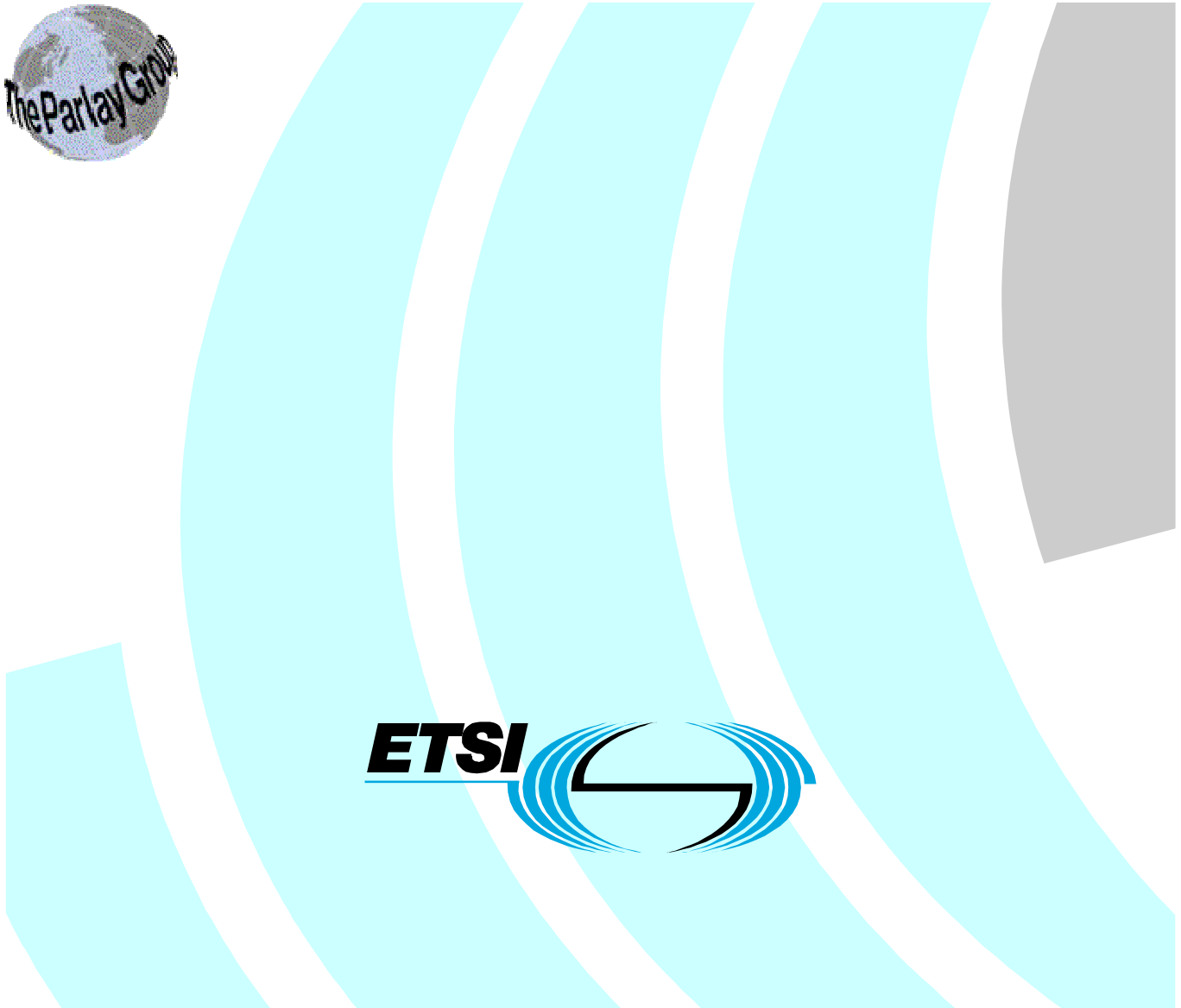
ETSI ES 203 915-5 V1.1.1 (2005-04)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF (Parlay 5)



Reference
DES/TISPAN-01005-05-OSA

Keywords
API, IDL, OSA, UML

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2005.
© The Parlay Group 2005.
All rights reserved.

DECT™, **PLUGTESTS**™ and **UMTS**™ are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Contents

Intellectual Property Rights 7
Foreword 7
1 Scope 8
2 References 8
3.1 Definitions 8
3.2 Abbreviations 8
4 Generic and Call User Interaction and Administration SCF 9
4.1 Generic and Call User Interaction SCF 9
4.2 Generic User Interaction Administration SCF 10
4.3 Generic User Interaction SCF Design Aspects 10
4.4 General requirements on support of methods 10
5 Sequence Diagrams 11
5.1 Generic and Call User Interaction Sequence Diagrams 11
5.1.1 Alarm Call 11
5.1.2 Call Barring 1 12
5.1.3 Network Controlled Notifications 13
5.1.4 Prepaid 14
5.1.5 Pre-Paid with Advice of Charge (AoC) 16
5.2 Generic User Interaction Administration Sequence Diagrams 19
5.2.1 Message Administration 19
6 Class Diagrams 20
6.1 Generic and Call User Interaction Class Diagrams 20
6.2 Generic User Interaction Administration Class Diagrams 21
7 The Service Interface Specifications 22
7.1 Interface Specification Format 22
7.1.1 Interface Class 22
7.1.2 Method descriptions 22
7.1.3 Parameter descriptions 22
7.1.4 State Model 22
7.2 Base Interface 22
7.2.1 Interface Class IpInterface 22
7.3 Service Interfaces 23
7.3.1 Overview 23
7.4 Generic Service Interface 23
7.4.1 Interface Class IpService 23
7.4.1.1 Method setCallback() 23
7.4.1.2 Method setCallbackWithSessionID() 23
8 Generic User Interaction Interface Classes 24
8.1 Generic and Call User Interaction Interface Classes 24
8.1.1 Interface Class IpUIManager 24
8.1.1.1 Method createUI() 25
8.1.1.2 Method createUICall() 25
8.1.1.3 Method createNotification() 25
8.1.1.4 Method destroyNotification() 26
8.1.1.5 Method changeNotification() 26
8.1.1.6 Method getNotification() 27
8.1.1.7 Method enableNotifications() 27
8.1.1.8 Method disableNotifications() 28
8.1.2 Interface Class IpAppUIManager 28
8.1.2.1 Method userInteractionAborted() 28
8.1.2.2 Method userInteractionNotificationInterrupted() 28
8.1.2.3 Method userInteractionNotificationContinued() 29

8.1.2.4 Method reportEventNotification()....29
8.1.2.5 Method <<new>> abortMultipleUserInteractions()....29
8.1.3 Interface Class IpUI....30
8.1.3.1 Method sendInfoReq()....30
8.1.3.2 Method sendInfoAndCollectReq()....31
8.1.3.3 Method release()....32
8.1.3.4 Method <<new>> setOriginatingAddress()....32
8.1.3.5 Method <<new>> getOriginatingAddress()....33
8.1.4 Interface Class IpAppUI....33
8.1.4.1 Method sendInfoRes()....33
8.1.4.2 Method sendInfoErr()....34
8.1.4.3 Method sendInfoAndCollectRes()....34
8.1.4.4 Method sendInfoAndCollectErr()....34
8.1.4.5 Method userInteractionFaultDetected()....35
8.1.5 Interface Class IpUICall....35
8.1.5.1 Method recordMessageReq()....35
8.1.5.2 Method deleteMessageReq()....36
8.1.5.3 Method abortActionReq()....36
8.1.5.4 Method <<new>> getMessageReq()....37
8.1.6 Interface Class IpAppUICall....37
8.1.6.1 Method recordMessageRes()....38
8.1.6.2 Method recordMessageErr()....38
8.1.6.3 Method deleteMessageRes()....38
8.1.6.4 Method deleteMessageErr()....39
8.1.6.5 Method abortActionRes()....39
8.1.6.6 Method abortActionErr()....39
8.1.6.7 Method <<new>> getMessageRes()....39
8.1.6.8 Method <<new>> getMessageErr()....40
8.2 Generic User Interaction Administration Interface Classes....40
8.2.1 Interface Class IpUIAdminManager....40
8.2.1.1 Method <<new>> getMessageReq()....40
8.2.1.2 Method <<new>> putMessageReq()....41
8.2.1.3 Method <<new>> deleteMessageReq()....41
8.2.1.4 Method <<new>> getMessageListReq()....42
8.2.2 Interface Class IpAppUIAdminManager....42
8.2.2.1 Method <<new>> getMessageRes()....43
8.2.2.2 Method <<new>> getMessageErr()....43
8.2.2.3 Method <<new>> deleteMessageRes()....44
8.2.2.4 Method <<new>> deleteMessageErr()....44
8.2.2.5 Method <<new>> putMessageRes()....44
8.2.2.6 Method <<new>> putMessageErr()....45
8.2.2.7 Method <<new>> getMessageListRes()....45
8.2.2.8 Method <<new>> getMessageListErr()....45

9 State Transition Diagrams....46
9.1 Generic and Call User Interaction State Transition Diagrams....46
9.1.1 State Transition Diagrams for IpUIManager....46
9.1.1.1 Active State....46
9.1.1.2 Notification Terminated State....46
9.1.2 State Transition Diagrams for IpUI....47
9.1.2.1 Active State....47
9.1.2.2 Release Pending State....47
9.1.2.3 Finished State....47
9.1.3 State Transition Diagrams for IpUICall....47
9.1.3.1 Active State....48
9.1.3.2 Release Pending State....48
9.1.3.3 Finished State....48
9.2 Generic User Interaction Administration State Transition Diagrams....49
9.2.1 State Transition Diagrams for IpUIAdminManager....49
9.2.1.1 Active State....49

10 Service Properties....49

10.1 User Interaction Service Properties 49
11 Data Definitions 50
11.1 TpUIFault 50
11.2 IpUI 50
11.3 IpUIRef 50
11.4 IpAppUI 50
11.5 IpAppUIRef 50
11.6 IpAppUIManager 50
11.7 IpAppUIManagerRef 50
11.8 TpUICallIdentifier 51
11.9 TpUICollectCriteria 51
11.10 TpUIError 52
11.11 TpUIEventCriteria 52
11.12 TpUIEventCriteriaResultSet 52
11.13 TpUIEventCriteriaResult 53
11.14 TpUIEventInfoDataType 53
11.15 TpUIIdentifier 53
11.16 TpUIIdentifierSet 53
11.17 TpUIInfo 53
11.18 TpUIInfoType 54
11.19 TpUIMessageCriteria 54
11.20 TpUIReport 55
11.21 TpUIResponseRequest 55
11.22 TpUITargetObjectType 55
11.23 TpUITargetObject 56
11.24 TpUIVariableInfo 56
11.25 TpUIVariableInfoSet 56
11.26 TpUIVariablePartType 56
11.27 TpUIEventNotificationInfo 57
11.28 TpUISynthesisInfoData 57
11.29 TpUISynthesisGender 57
11.30 TpUISynthesisAge 57
11.31 TpUISynthesisRate 58
11.32 TpUISynthesisRange 58
11.33 TpUIWordOverrideSet 58
11.34 TpUIWordOverride 58
11.35 TpUIPronounceType 58
11.36 TpUICollectMode 59
11.37 TpUIRecognitionCriteria 59
11.38 TpUIRecognitionSpeakerID 59
11.39 TpUIRecognitionPropertySet 59
11.40 TpUIRecognitionProperty 59
11.41 TpUIRecognitionGrammar 60
11.42 TpMessageIDList 61
12 Exception Classes 61
Annex A (normative): OMG IDL Description of User Interaction SCF 62
Annex B (informative): W3C WSDL Description of User Interaction SCF 63
Annex C (informative): Java™ API Description of the User Interaction SCF 64
Annex D (informative): Contents of 3GPP OSA R6 User Interaction 65
Annex E (informative): Description of Generic User Interaction SCF for 3GPP2 cdma2000 networks 66
E.1 General Exceptions 66
E.2 Specific Exceptions 66
E.2.1 Clause 1: Scope 66
E.2.2 Clause 2: References 66

E.2.3 Clause 3: Definitions and abbreviations 66
E.2.4 Clause 4: Generic and Call User Interaction SCF 66
E.2.5 Clause 5: Class Diagrams 66
E.2.6 Clause 6: Class Diagrams 66
E.2.7 Clause 7: The Service Interface Specifications 67
E.2.8 Clause 8: Generic User Interaction Interface Classes Definitions 67
E.2.9 Clause 9: State Transition Diagrams 67
E.2.10 Clause 10: Service Properties 67
E.2.11 Clause 11: Data Definitions 67
E.2.12 Clause 12: Exception Classes 67
E.2.13 Annex A (normative): OMG IDL Description of Generic User Interaction SCF 67
E.2.14 Annex B (informative): W3C WSDL Description of Generic User Interaction SCF 67
E.2.15 Annex C (informative): Java™ API Description of Generic User Interaction SCF 67

Annex F (informative): Record of changes 68

F.1 Interfaces 68
F.1.1 New 68
F.1.2 Deprecated 68
F.1.3 Removed 68

F.2 Methods 69
F.2.1 New 69
F.2.2 Deprecated 69
F.2.3 Modified 69
F.2.4 Removed 69

F.3 Data Definitions 70
F.3.1 New 70
F.3.2 Modified 70
F.3.3 Removed 70

F.4 Service Properties 71
F.4.1 New 71
F.4.2 Deprecated 71
F.4.3 Modified 71
F.4.4 Removed 71

F.5 Exceptions 71
F.5.1 New 71
F.5.2 Modified 71
F.5.3 Removed 72

F.6 Others 72

History 73

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 5 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 203 915) is structured in the following parts:

Part 1: "Overview";

Part 2: "Common Data Definitions";

Part 3: "Framework";

Part 4: "Call Control";

Part 5: "User Interaction SCF";

Part 6: "Mobility SCF";

Part 7: "Terminal Capabilities SCF";

Part 8: "Data Session Control SCF";

Part 9: "Generic Messaging SCF";

Part 10: "Connectivity Manager SCF";

Part 11: "Account Management SCF";

Part 12: "Charging SCF";

Part 13: "Policy Management SCF";

Part 14: "Presence and Availability Management SCF";

Part 15: "Multi-Media Messaging SCF".

The present document has been defined jointly between ETSI, The Parlay Group (http://www.parlay.org) and the 3GPP, in co-operation with a number of JAIN™ Community (http://www.java.sun.com/products/jain) member companies.

The present document forms part of the Parlay 5.0 set of specifications.

The present document is equivalent to 3GPP TS 29.198-5 V6.2.0 (Release 6).

1 Scope

The present document is part 5 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs.

The present document specifies the User Interaction (UI) Service Capability Feature (SCF) aspects of the interface. All aspects of the User Interaction SCF are defined here, these being:

- Sequence Diagrams.
- Class Diagrams.
- Interface specification plus detailed method descriptions.
- State Transition diagrams.
- Data Definitions.
- IDL Description of the interfaces.
- WSDL Description of the interfaces.
- Reference to the Java™ API description of the interfaces.

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

2 References

The references listed in clause 2 of ES 203 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 203 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 5)".

ETSI ES 203 915-2: "Open Service Access (OSA); Application Programming Interface (API); Part 2: Common Data Definitions (Parlay 5)".

ETSI ES 203 915-4-2: "Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 2: Generic Call Control SCF (Parlay 5)".

ETSI ES 203 915-4-3: "Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 5)".

World Wide Web Consortium: "Speech Recognition Grammar Specification Version 1"
http://www.w3.org/TR/2004/REC-speech-grammar-20040316/

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 203 915-1 apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 203 915-1 apply.

4 Generic and Call User Interaction and Administration SCF

4.1 Generic and Call User Interaction SCF

The Generic User Interaction service capability feature is used by applications to interact with end users. It consists of three interfaces:

1) User Interaction Manager, containing management functions for User Interaction related issues.

2) Generic User Interaction, containing methods to interact with an end-user.

3) Call User Interaction, containing methods to interact with an end-user engaged in a call.

The Generic User Interaction service capability feature is described in terms of the methods in the Generic User Interaction interfaces.

Table 1 gives an overview of the Generic User Interaction methods and to which interfaces these methods belong.

Table 1: Overview of Generic User Interaction interfaces and their methods

User Interaction Manager	Generic User Interaction
createUI	sendInfoReq
createUICall	sendInfoRes
createNotification	sendInfoErr
destroyUINotification	sendInfoAndCollectReq
reportEventNotification	sendInfoAndCollectRes
userInteractionAborted	sendInfoAndCollectErr
userInteractionNotificationInterrupted	release
userInteractionNotificationContinued	userInteractionFaultDetected
changeNotification	setOriginatingAddress
getNotification	getOriginatingAddress
enableNotifications	
disableNotifications	
abortMultipleUserInteractions	

Table 2 gives an overview of the Call User Interaction methods and to which interfaces these methods belong.

Table 2: Overview of Call User Interaction interfaces and their methods

User Interaction Manager	Call User Interaction
As defined for the Generic User Interaction SCF	Inherits from Generic User Interaction and adds:
	recordMessageReq
	recordMessageRes
	recordMessageErr
	deleteMessageReq
	deleteMessageRes
	deleteMessageErr
	abortActionReq
	abortActionRes
	abortActionErr
	getMessageReq
	getMessageRes
	getMessageErr

The IpUI Interface provides functions to send information to, or gather information from the user, i.e. this interface allows applications to send SMS and USSD messages. An application can use this interface independently of other SCFs. The IpUICall Interface provides functions to send information to, or gather information from the user (or call party) attached to a call.

4.2 Generic User Interaction Administration SCF

The Generic User Interaction Administration service capability feature is used by application to interact with the service to manage the user announcement and recorded messages. It consists of one interface:

1) User Interaction Administration Manager, containing message management functions for User Interaction.

Table 3: Overview of Generic User Interaction Administration interfaces and their methods

User Interaction Administration Manager
getMessageReq
putMessageReq
deleteMessageReq
getMessageListReq

4.3 Generic User Interaction SCF Design Aspects

The following clauses describe each aspect of the Generic User Interaction and Generic User Interaction Administration Service Capability Features (SCF).

The order is as follows:

- The Sequence diagrams give the reader a practical idea of how each of the SCFs is implemented.
- The Class relationships clause shows how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part. This clause also includes Call User interaction.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause shows a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 203 915-2.

4.4 General requirements on support of methods

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method.

Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

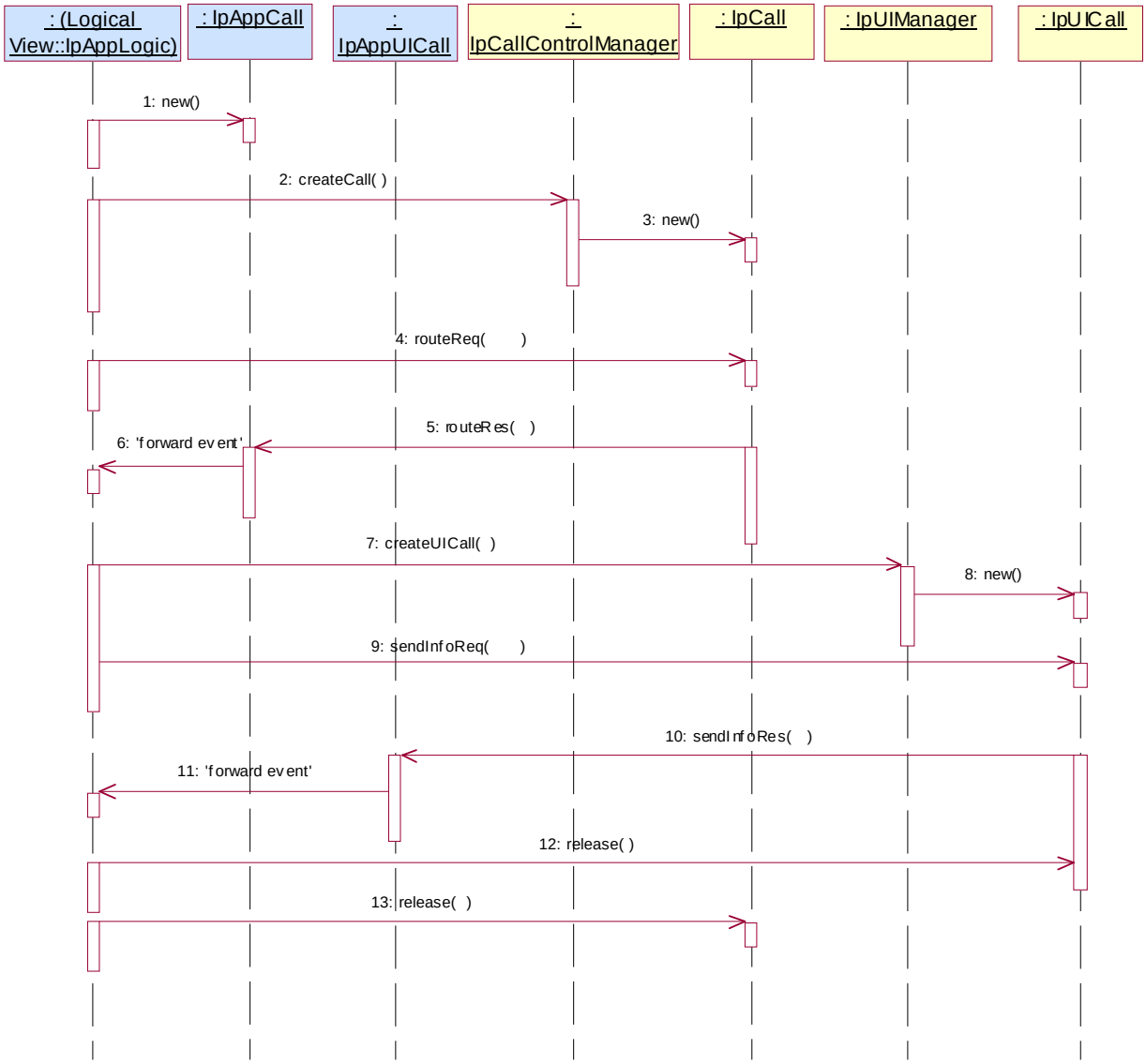
Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

5 Sequence Diagrams

5.1 Generic and Call User Interaction Sequence Diagrams

5.1.1 Alarm Call

The following sequence diagram shows a 'reminder message', in the form of an alarm, being delivered to a customer as a result of a trigger from an application. Typically, the application would be set to trigger at a certain time, however, the application could also trigger on events.



1: This message is used to create an object implementing the IpAppCall interface.

2: This message requests the object implementing the IpCallControlManager interface to create an object implementing the IpCall interface.

3: Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met it is created.

4: This message instructs the object implementing the IpCall interface to route the call to the customer destined to receive the 'reminder message'.

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

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

7: The application requests a new UICall object that is associated with the call object.

8: Assuming all criteria are met, a new UICall object is created by the service.

9: This message instructs the object implementing the IpUICall interface to send the alarm to the customer's call.

10: When the announcement ends this is reported to the call back interface.

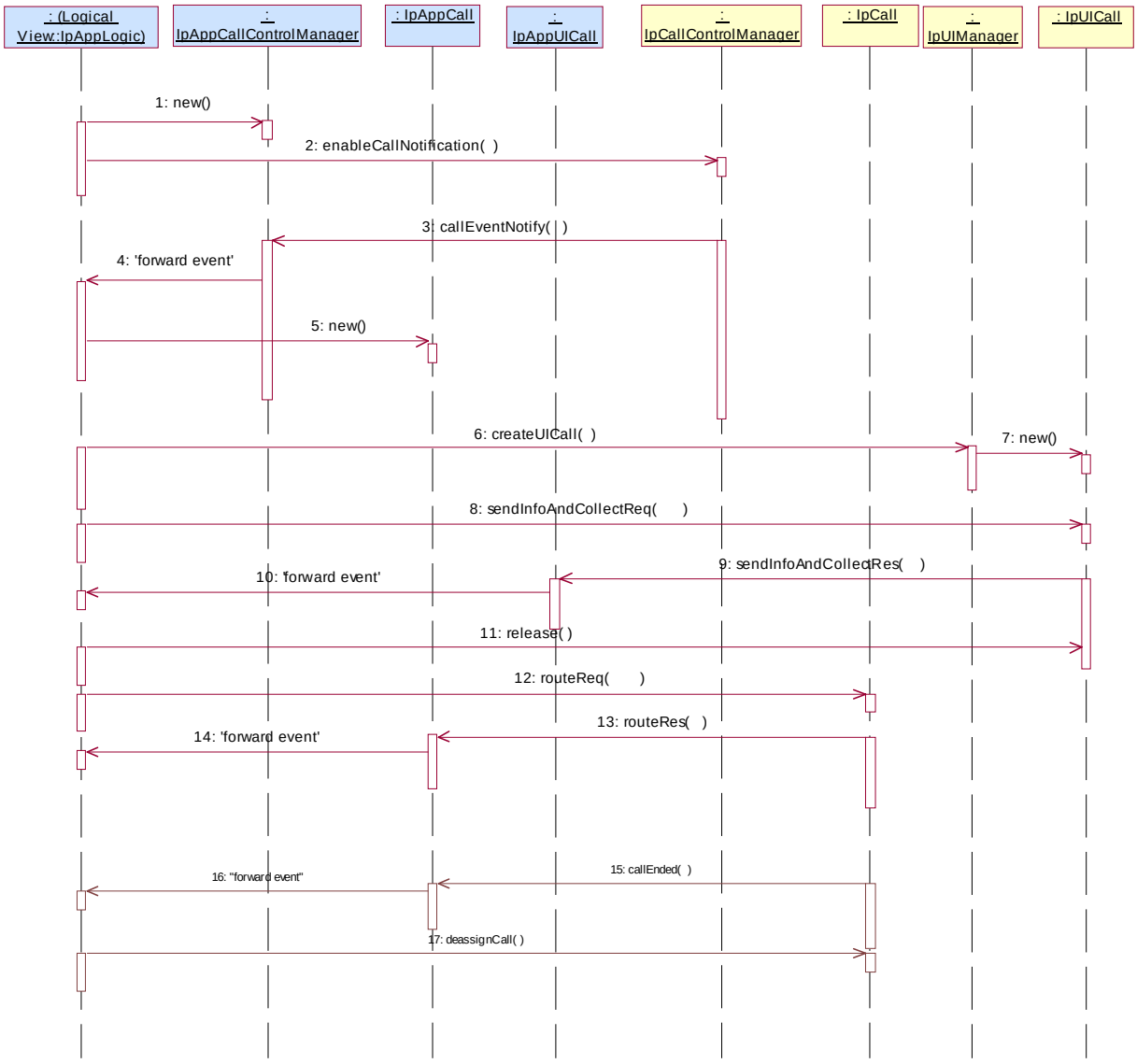
11: The event is forwarded to the application logic.

12: The application releases the UICall object, since no further announcements are required. Alternatively, the application could have indicated P_FINAL_REQUEST in the sendInfoReq in which case the UICall object would have been implicitly released after the announcement was played.

13: The application releases the call and all associated parties.

5.1.2 Call Barring 1

The following sequence diagram shows a call barring service, initiated as a result of a prearranged event being received by the 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

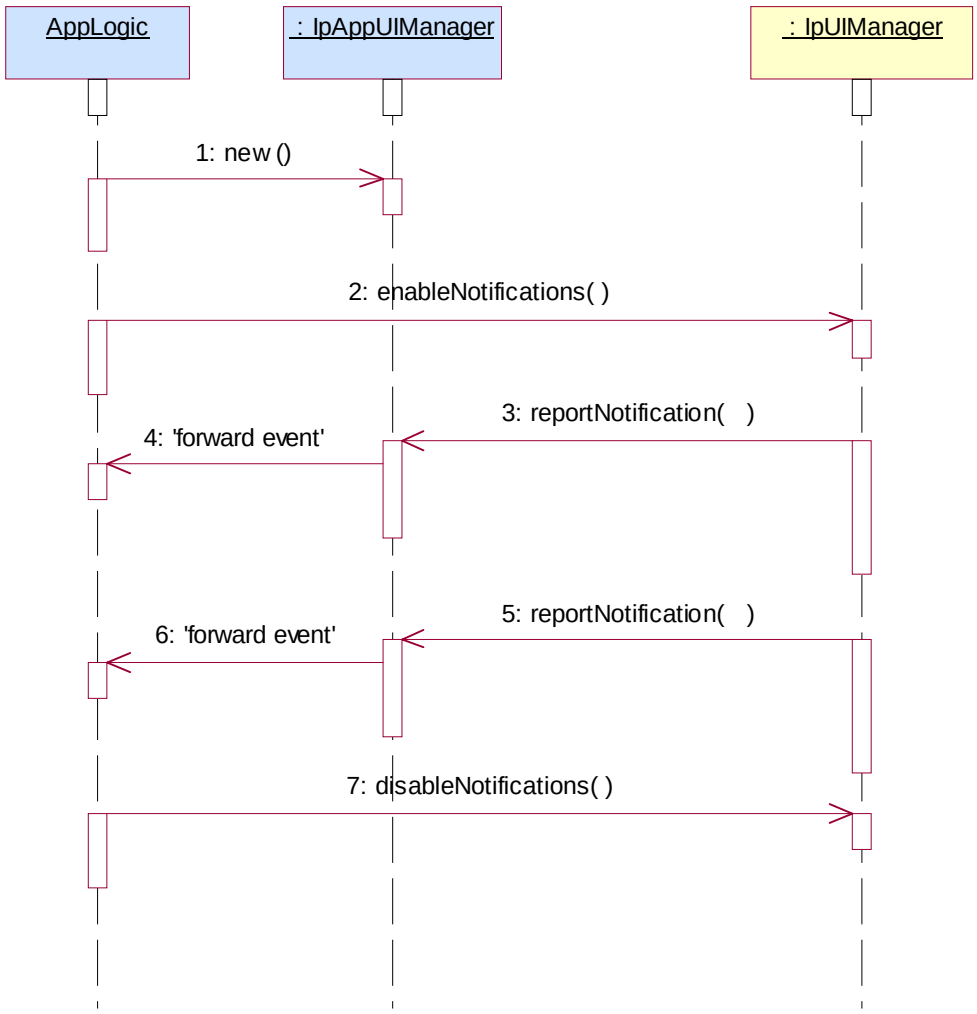
15: When the call is terminated in the network, the application will receive a notification. This notification will always be received when the call is terminated by the network in a normal way, the application does not have to request this event explicitly.

16: The event is forwarded to the application.

17: The application must free the call related resources in the gateway by calling deassignCall.

5.1.3 Network Controlled Notifications

The following sequence diagram shows how an application can receive notifications that have not been created by the application, but are provisioned from within the network.



1: The application is started. The application creates a new IpAppUIManager to handle callbacks.

2: The enableNotifications method is invoked on the IpUIManager interface to indicate that the application is ready to receive notifications that are created in the network. For illustrative purposes we assume notifications of type "B" are created in the network.

3: When a network created trigger occurs the application is notified on the callback interface.

4: The event is forwarded to the application.

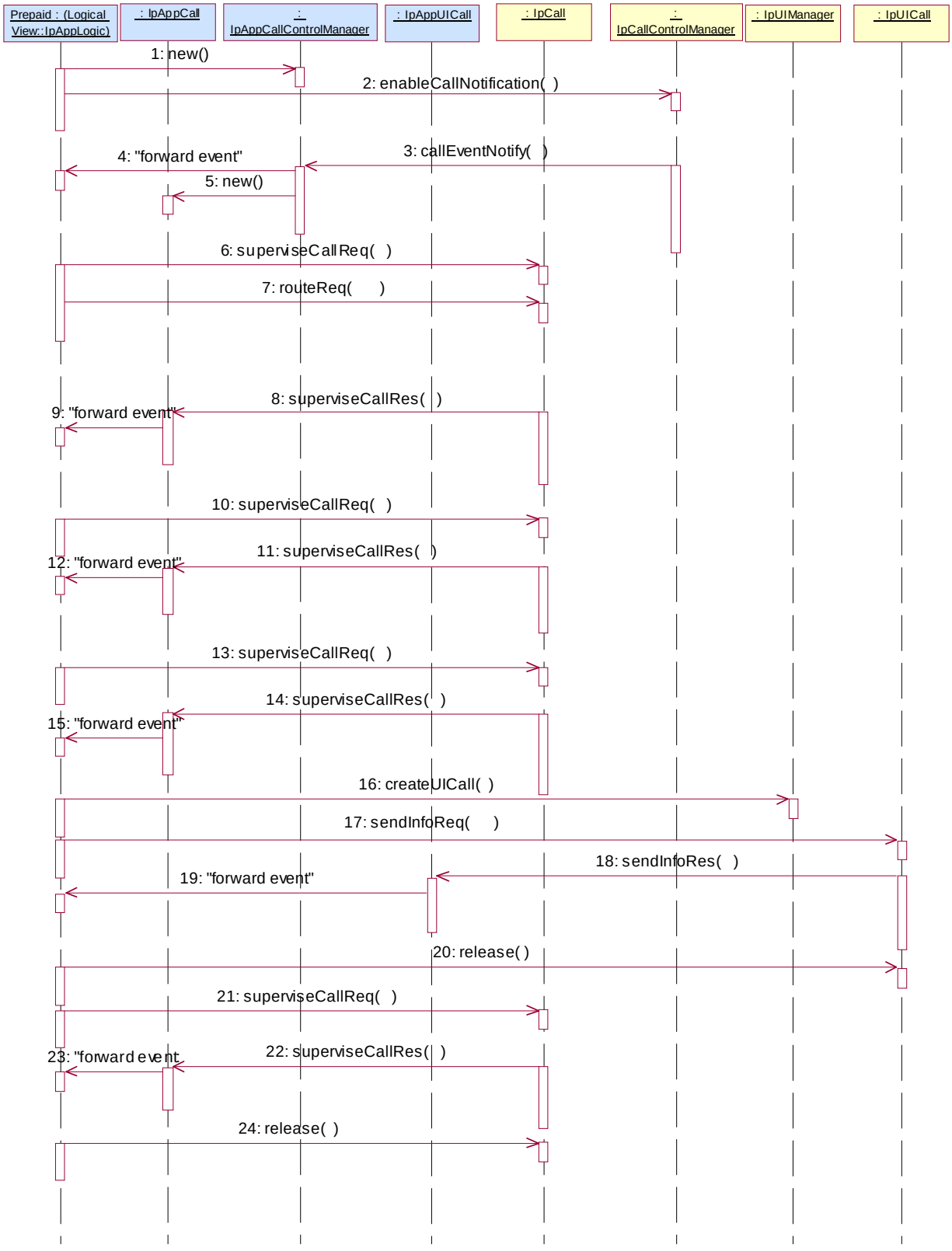
5: When a network created trigger occurs the application is notified on the callback interface.

6: The event is forwarded to the application.

7: When the application does not want to receive notifications created in the network anymore, it invokes disableNotifications on the IpMultiPartyCallConrolManager interface. From now on the gateway will not send any notifications to the application that are created in the network.

5.1.4 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.

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 call.

17: An announcement is played informing the user about the near-expiration of his credit limit.

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.

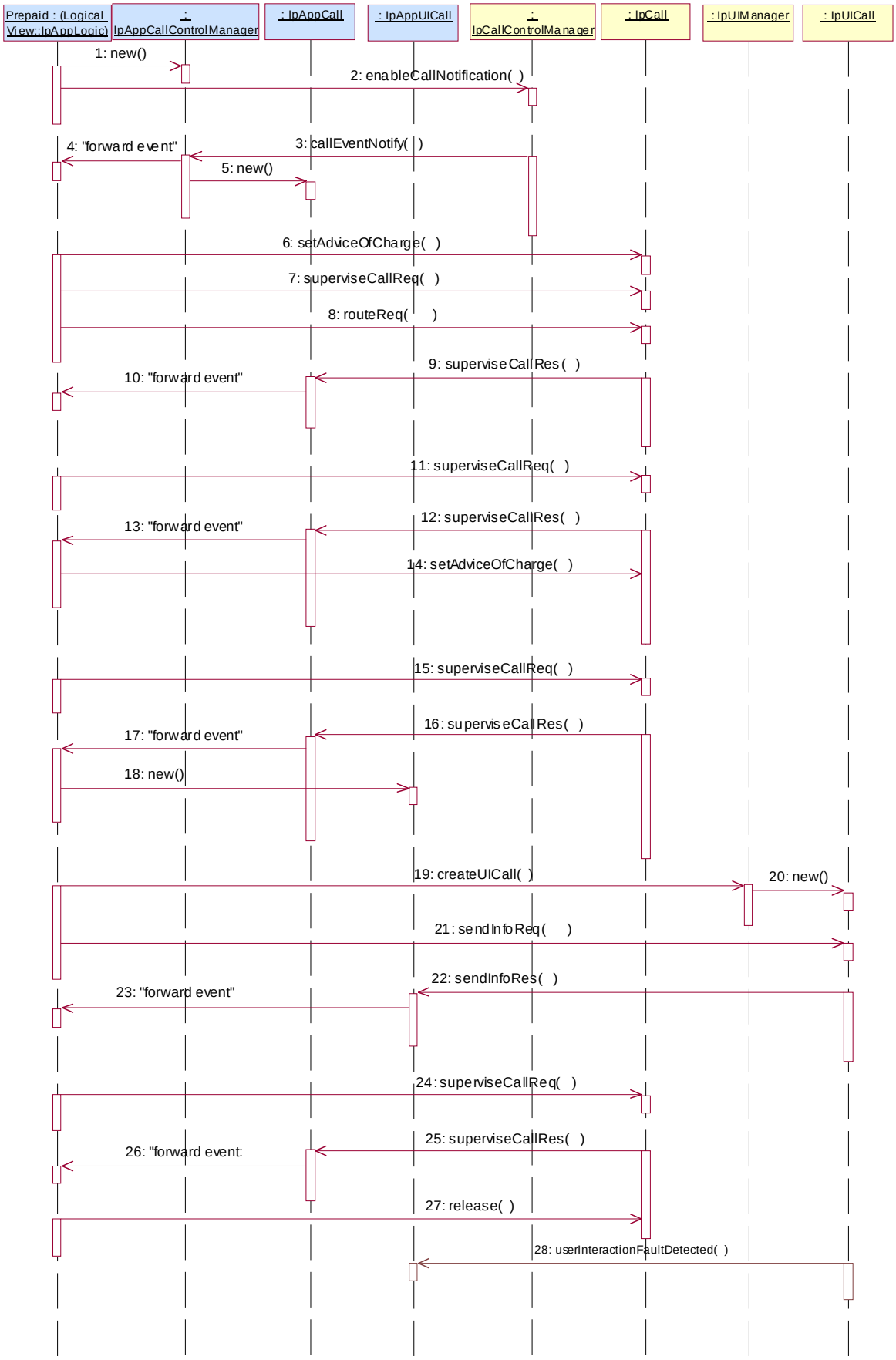
22: The supervision period ends.

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.

5.1.5 Pre-Paid with Advice of Charge (AoC)

This sequence shows a Pre-paid application that uses the Advice of Charge feature. The application will send the charging information before the actual call setup and when during the call the charging changes new information is sent in order to update the end-user. Note that the Advice of Charge feature requires an application in the end-user terminal to display the charges for the call, depending on the information received from the application.


Prepaid : (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpAppUICall
: IpCallControlManager
: IpCall
: IpUIManager
: IpUICall
1: new()
2: enableCallNotification( )
3: callEventNotify( )
4: "forward event"
5: new()
6: setAdviceOfCharge( )
7: superviseCallReq( )
8: routeReq( )
9: superviseCallRes( )
10: "forward event"
11: superviseCallReq( )
12: superviseCallRes( )
13: "forward event"
14: setAdviceOfCharge( )
15: superviseCallReq( )
16: superviseCallRes( )
17: "forward event"
18: new()
19: createUICall( )
20: new()
21: sendInfoReq( )
22: sendInfoRes( )
23: "forward event"
24: superviseCallReq( )
25: superviseCallRes( )
26: "forward event:
27: release( )
28: userInteractionFaultDetected( )

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 Call object is created.

6: The Pre-Paid Application (PPA) sends the AoC information (e.g. the tariff switch time). (it shall be noted the PPA contains ALL the tariff information and knows how to charge the user).

During this call sequence 2 tariff changes take place. The call starts with tariff 1, and at the tariff switch time (e.g. 18:00 hours) switches to tariff 2. The application is not informed about this (but the end-user is!).

7: 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.

8: The application requests to route the call to the destination address.

9: At the end of each supervision period the application is informed and a new period is started.

10: The message is forwarded to the application.

11: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

12: At the end of each supervision period the application is informed and a new period is started.

13: The message is forwarded to the application.

14: Before the next tariff switch (e.g. 19:00 hours) the application sends a new AOC with the tariff switch time. Again, at the tariff switch time, the network will send AoC information to the end-user.

15: 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.

16: When the user is almost out of credit the application is informed.

17: The message is forwarded to the application.

18: The application creates a new call back interface for the User interaction messages.

19: A new UI Call object that will handle playing of the announcement needs to be created.

20: The Gateway creates a new UI call object that will handle playing of the announcement.

21: With this message the announcement is played to the parties in the call.

22: The user indicates that the call should continue.

23: The message is forwarded to the application.

24: The user does not terminate so the application terminates the call after the next supervision period.

25: The user is out of credit and the application is informed.

26: The message is forwarded to the application.

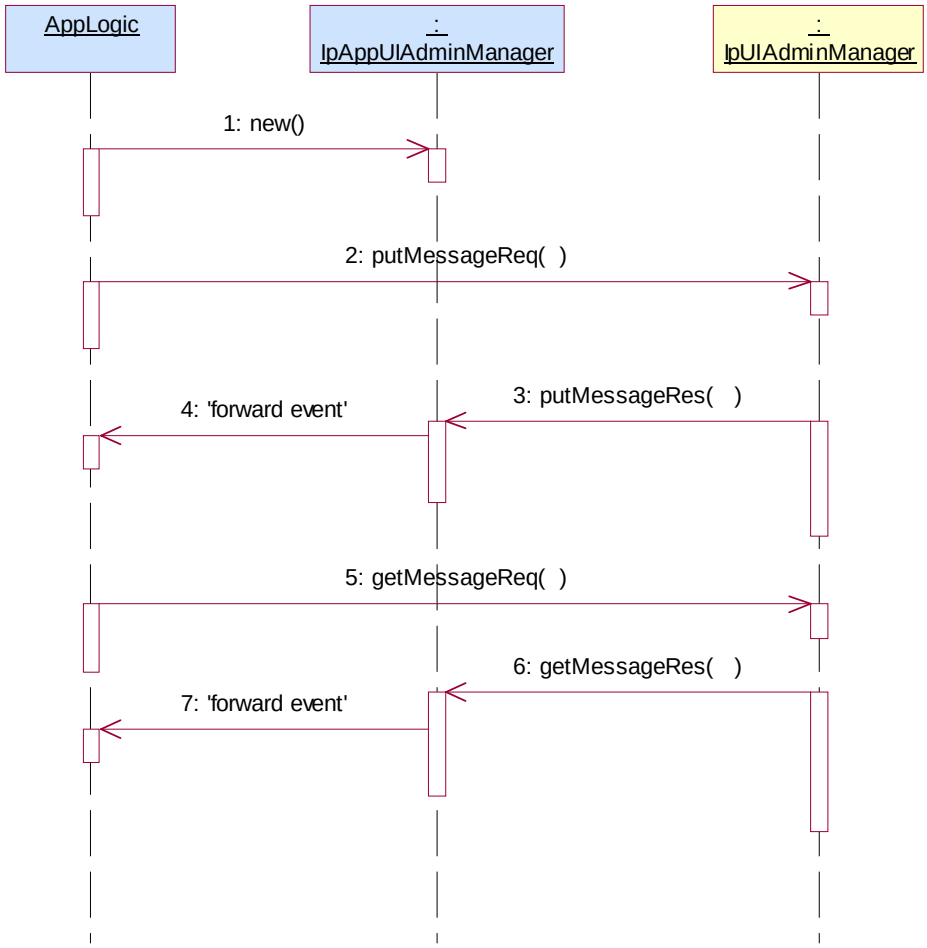
27: With this message the application requests to release the call.

28: Terminating the call which has still a UICall object associated will result in a userInteractionFaultDetected. The UICall object is terminated in the gateway and no further communication is possible between the UICall and the application.

5.2 Generic User Interaction Administration Sequence Diagrams

5.2.1 Message Administration

The following sequence diagram shows how an application can manage the user announcement and recorded messages.



1: The application is started. The application creates a new IpAppUIAdminManager to handle callbacks.

2: The putMessageReq method is invoked on the IpUIAdminManager interface to create a new pre-defined message for use by sending to the user.

3: The putMessageRes response notifies the application of the messageID on the callback interface.

4: The response is forwarded to the application logic.

5: The getMessageReq method is invoked on the IpUIAdminManager interface to retrieve the contents of a user announcement or recorded message.

6: The getMessageRes response notifies the application of the contents of a message.

7: The event is forwarded to the application.

6 Class Diagrams

6.1 Generic and Call User Interaction Class Diagrams

The application generic user interaction service package consists of one IpAppUIManager interface, zero or more IpAppUI interfaces and zero or more IpAppUICall interfaces.

The generic user interaction service package consists of one IpUIManager interface, zero or more IpUI interfaces and zero or more IpUICall interfaces.

The class diagram in the following figure shows the interfaces that make up the application generic user interaction service package and the generic user interaction service package. Communication between these packages is done via the <<uses>> relationships.

The IpUICall implements call related user interaction and it inherits from the non call related IpUI interface. The same holds for the corresponding application interfaces.

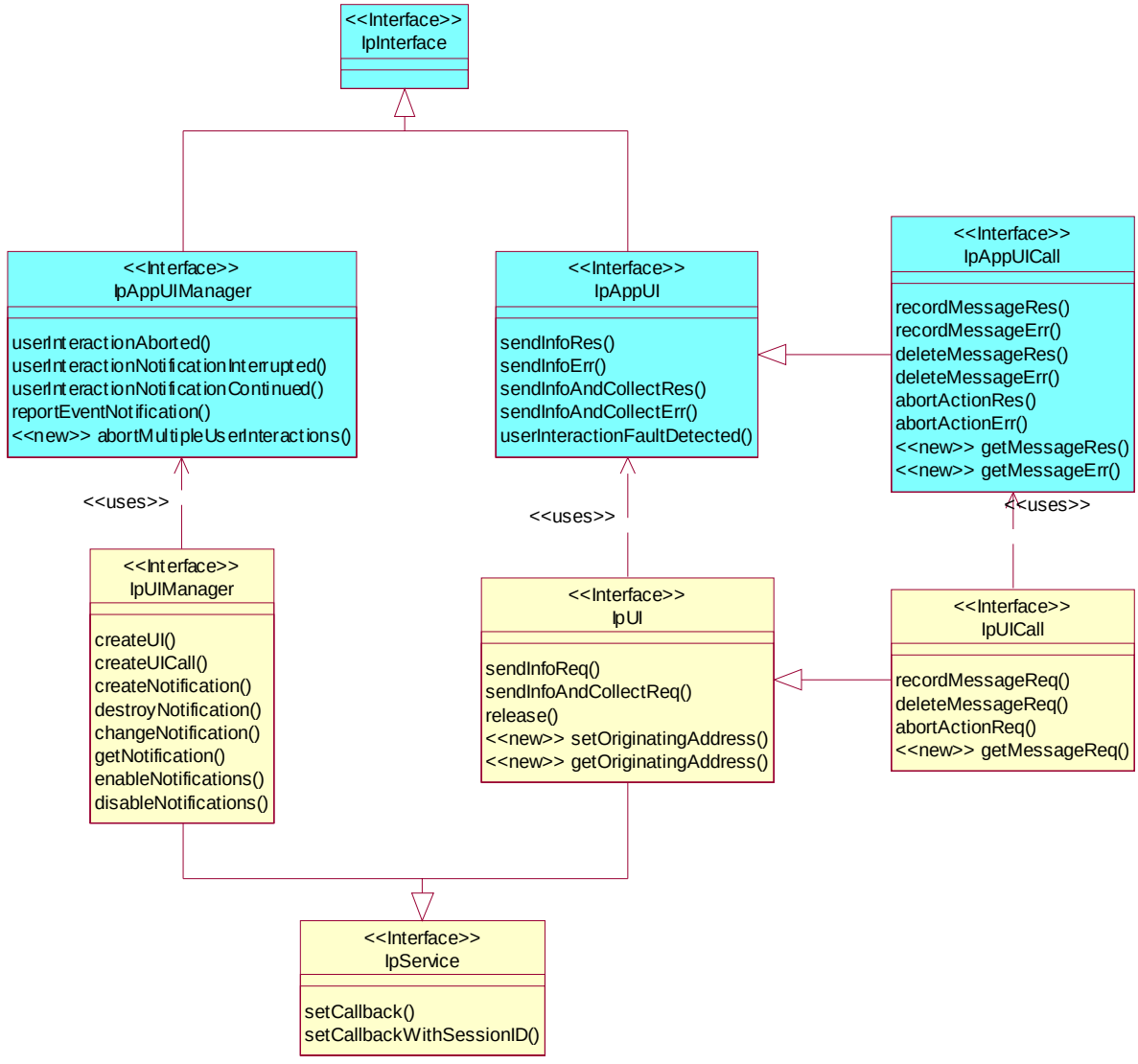


Figure 1: Generic User Interaction Package Overview

6.2 Generic User Interaction Administration Class Diagrams

The application generic user administration service package consists of one IpAppUIAdminManager interface and one IpUIAdminManager interfaces.

The class diagram in the following figure shows the interfaces that make up the application generic user administration service package. Communication between these packages is done via the <<uses>> relationships.

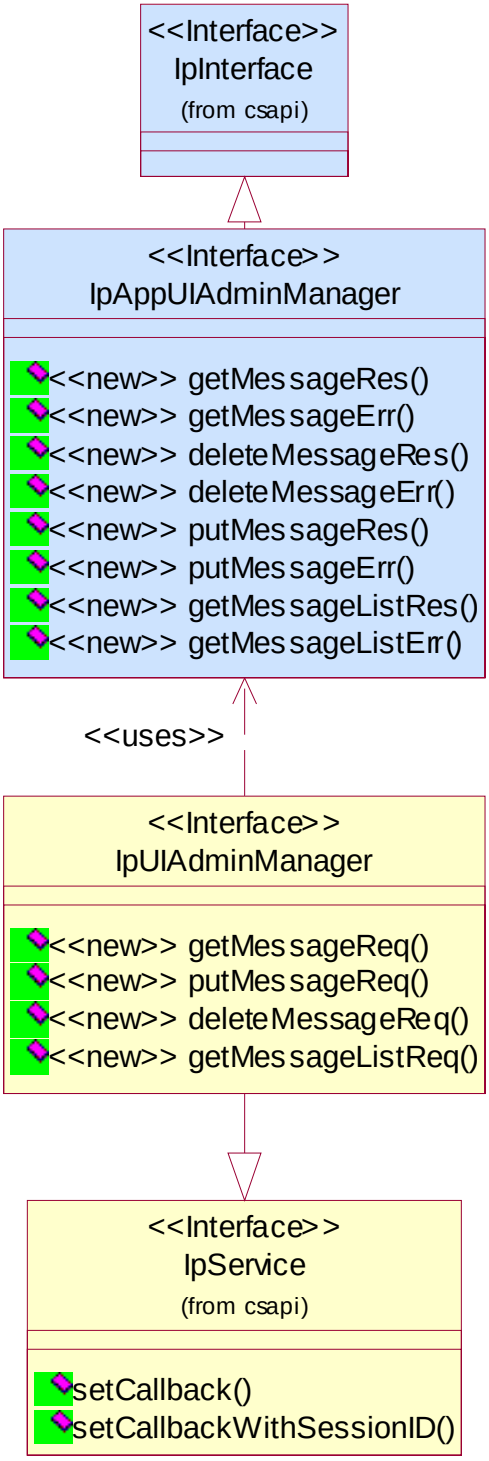


Figure 2: Generic User Administration Package Overview

7 The Service Interface Specifications

7.1 Interface Specification Format

This clause defines the interfaces, methods and parameters that form a part of the API specification. The Unified Modelling Language (UML) is used to specify the interface classes. The general format of an interface specification is described below.

7.1.1 Interface Class

This shows a UML interface class description of the methods supported by that interface, and the relevant parameters and types. The Service and Framework interfaces for enterprise-based client applications are denoted by classes with name `Ip<name>`. The callback interfaces to the applications are denoted by classes with name `IpApp<name>`. For the interfaces between a Service and the Framework, the Service interfaces are typically denoted by classes with name IpSvc<name>, while the Framework interfaces are denoted by classes with name IpFw<name>.

7.1.2 Method descriptions

Each method (API method 'call') is described. Both synchronous and asynchronous methods are used in the API. Asynchronous methods are identified by a '`Req`' suffix for a method request, and, if applicable, are served by asynchronous methods identified by either a '`Res`' or '`Err`' suffix for method results and errors, respectively. To handle responses and reports, the application or service developer must implement the relevant `IpApp<name> or IpSvc<name>` interfaces to provide the callback mechanism.

7.1.3 Parameter descriptions

Each method parameter and its possible values are described. Parameters described as 'in' represent those that must have a value when the method is called. Those described as 'out' are those that contain the return result of the method when the method returns.

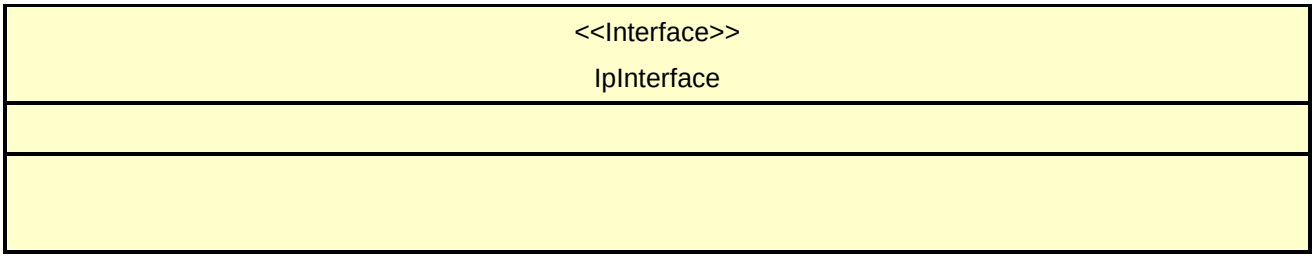
7.1.4 State Model

If relevant, a state model is shown to illustrate the states of the objects that implement the described interface.

7.2 Base Interface

7.2.1 Interface Class IpInterface

All application, framework and service interfaces inherit from the following interface. This API Base Interface does not provide any additional methods.



7.3 Service Interfaces

7.3.1 Overview

The Service Interfaces provide the interfaces into the capabilities of the underlying network - such as call control, user interaction, messaging, mobility and connectivity management.

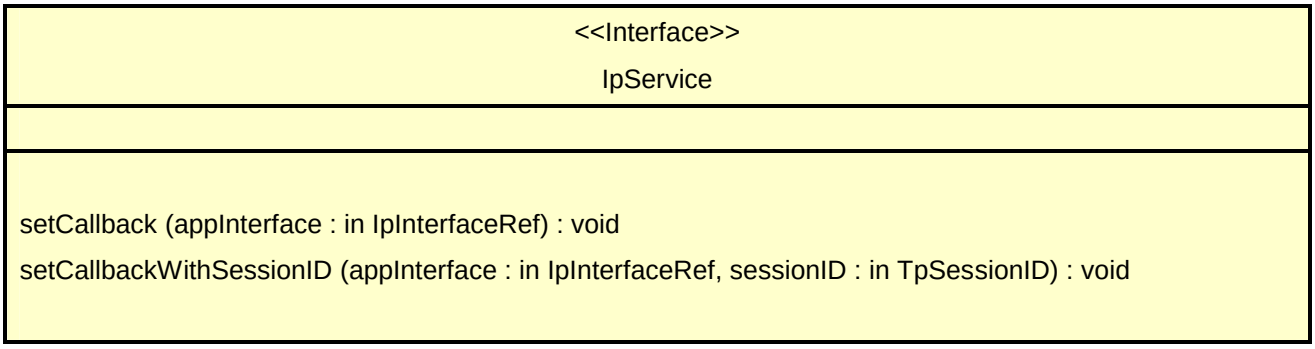
The interfaces that are implemented by the services are denoted as 'Service Interface'. The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as 'Application Interface'.

7.4 Generic Service Interface

7.4.1 Interface Class IpService

Inherits from: IpInterface.

All service interfaces inherit from the following interface.



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

8 Generic User Interaction Interface Classes

8.1 Generic and Call User Interaction Interface Classes

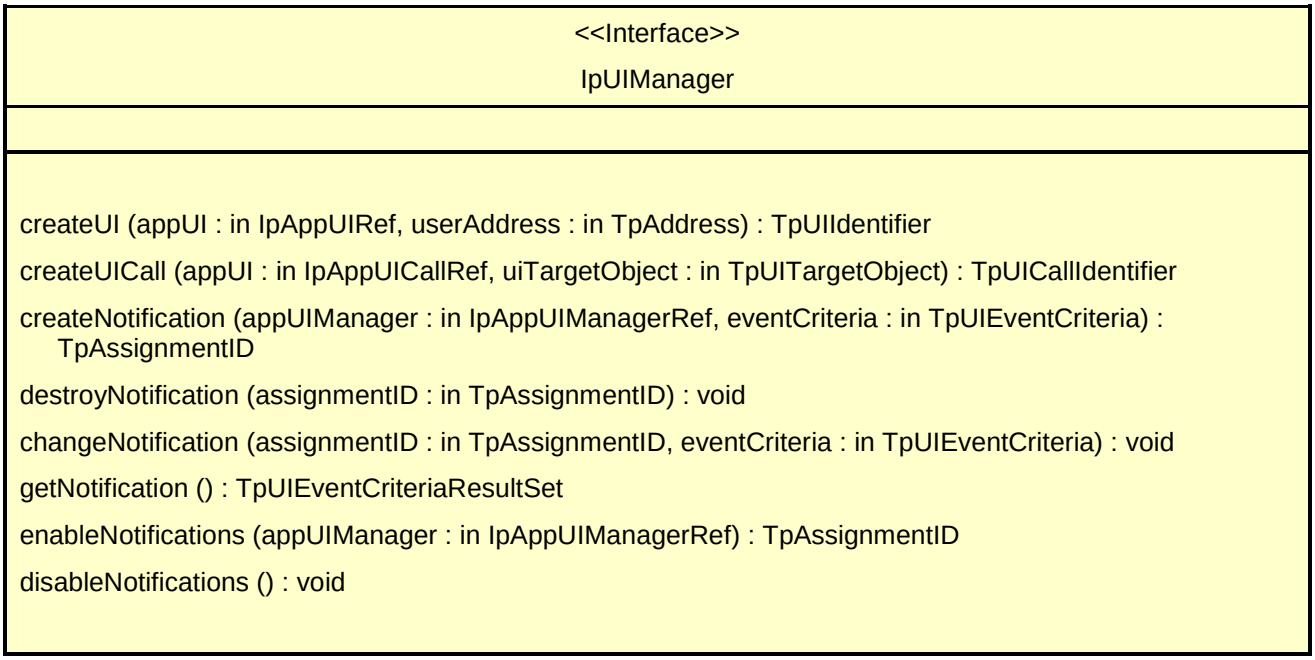
The Generic User Interaction Service interface (GUIS) is used by applications to interact with end users. The GUIS is represented by the IpUIManager, IpUI and IpUICall interfaces that interface to services provided by the network. To handle responses and reports, the developer must implement IpAppUIManager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

8.1.1 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.



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 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 gateway 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 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 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 gateway 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 reportEventNotification() 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 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.

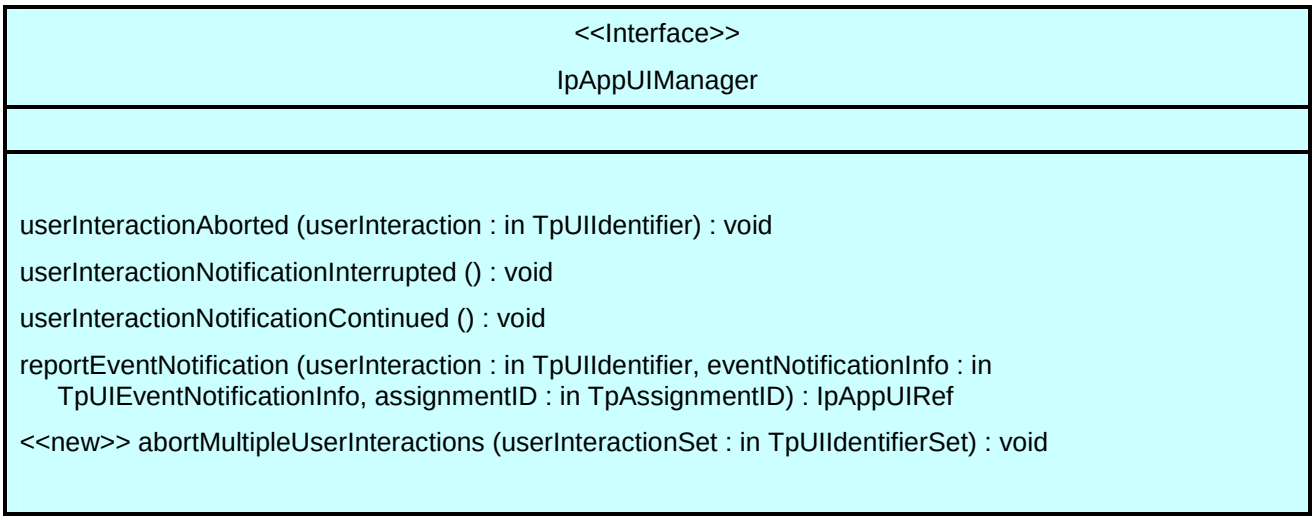
Raises

`TpCommonExceptions`

8.1.2 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.



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 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.

8.1.2.3 Method userInteractionNotificationContinued()

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

Parameters
No Parameters were identified for this method.

8.1.2.4 Method 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`

8.1.2.5 Method <<new>> 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`

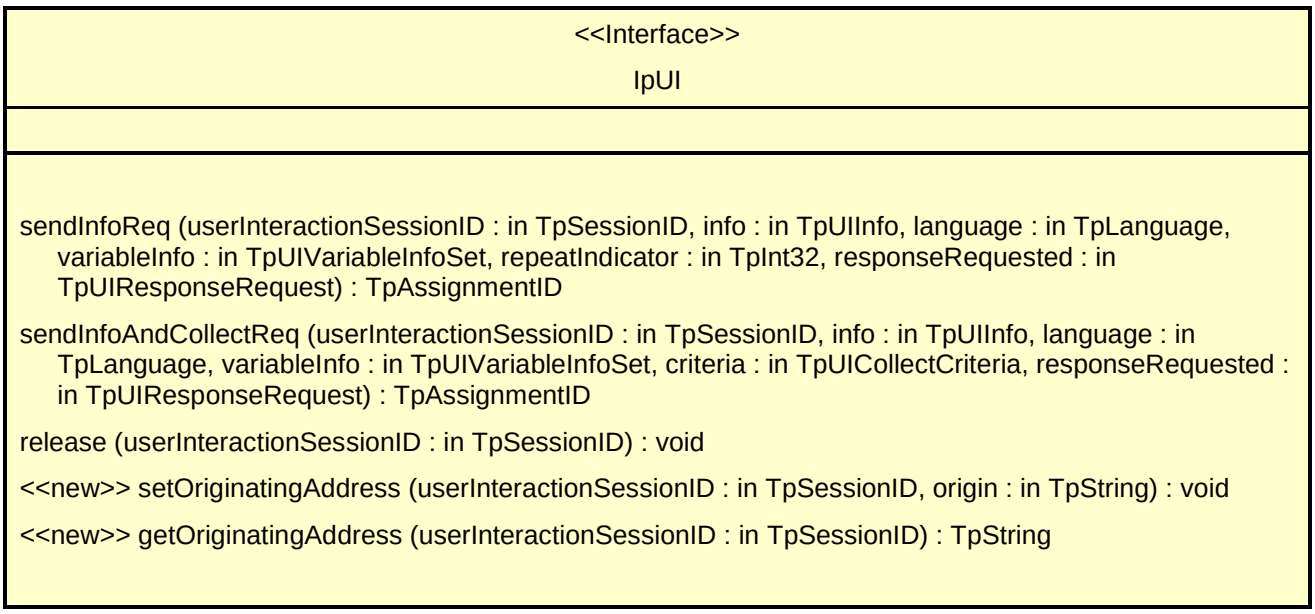
Specifies the set of interfaces and sessionIDs of the user interaction sessions that have aborted or terminated abnormally. The empty set shall be used to indicate that all user interactions have aborted.

8.1.3 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.

This interface, or the IpUICall interface, shall be implemented by a Generic User Interaction SCF as a minimum requirement. The release() method, and at least one of the sendInfoReq() or the sendInfoAndCollectReq() methods shall be implemented as a minimum requirement.



8.1.3.1 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 sent (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. A URL enables the application to utilize dynamic multi-media content by reference;

- Binary Data, identifying pre-defined information or data to be sent to or downloaded into the terminal. Binary data enables the application to utilize dynamic multi-media content directly;

- a VXML string defines the Voice XML page to execute on the server and interact with the end-user. The VXML page execution continues until an <exit/> tag is encountered, which results in a sendInfoRes() callback;

- a SynthesisInfo structure defines the text to synthesize and how the synthesis should be done.

language : in TpLanguage

Specifies the Language of the information to be sent 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

8.1.3.2 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 sent (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. A URL enables the application to utilize dynamic multi-media content by reference;

- Binary Data, identifying pre-defined information or data to be sent to or downloaded into the terminal. Binary data enables the application to utilize dynamic multi-media content directly;

- a VXML string defines the Voice XML page to execute on the server and interact with the end-user. The VXML page execution continues until an <exit/> tag is encountered, which results in a sendInfoAndCollectRes() callback with the value of the expr= attribute;

- a SynthesisInfo structure defines the text to synthesize and how the synthesis should be done.

language : in TpLanguage

Specifies the Language of the information to be sent 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. This parameter also specifies whether voice recognition would be used.

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. If P_UI_RESPONSE_REQUIRED is not enabled by the application request, the user interaction shall nevertheless return either a sendInfoAndCollectRes or sendInfoAndCollectErr method to the application in response to this method invocation.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE, P_ILLEGAL_ID, P_ID_NOT_FOUND, P_ILLEGAL_RANGE, P_INVALID_COLLECTION_CRITERIA

8.1.3.3 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

8.1.3.4 Method <<new>> setOriginatingAddress()

This method sets the originating address property on the user interaction session to be used when sending information to the user.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

origin : in TpString

Specifies the originating address. The originating address description is sent as a TpString. However this field may contain E.164 addresses that the receiving terminal can use to reply to the message. The coding of such an E.164 address can either be local numbers or international numbers, according to the standard E.164. Examples for a local number is "0702106181" and for an international number "+46702106181".

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS

8.1.3.5 Method <<new>> getOriginatingAddress()

This method gets the originating address property on the user interaction session to be used when sending information to the user. If not set with setOriginatingAddress(), the getOriginatingAddress() returns the description that would be displayed on the terminal device as the originating address when a message is sent with sendInfoReq() or sendInfoAndCollectReq().

Returns: TpString.

The address that will be used for a sendInfoReq() or sendInfoAndCollectReq() for the originating address.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

Returns

TpString

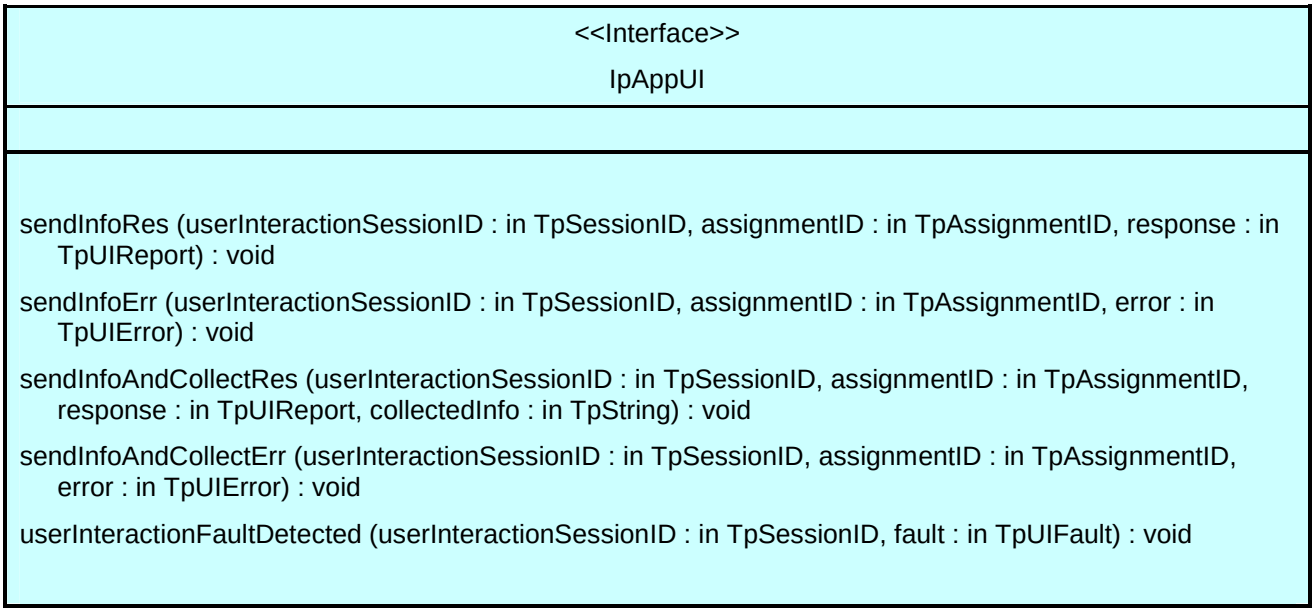
Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

8.1.4 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.



8.1.4.1 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_UI_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.

8.1.4.2 Method sendInfoErr()

This asynchronous method indicates that the request to send information was unsuccessful. This response is called only if the responseRequested parameter of the sendInfoReq() method was set to P_UI_RESPONSE_REQUIRED. In the event that a response was not requested and the user interaction was unsuccessful the implementation of the service capability must handle the network error, however the error shall not be reported to the application as it requested no response.

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.

8.1.4.3 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.

8.1.4.4 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.

8.1.4.5 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

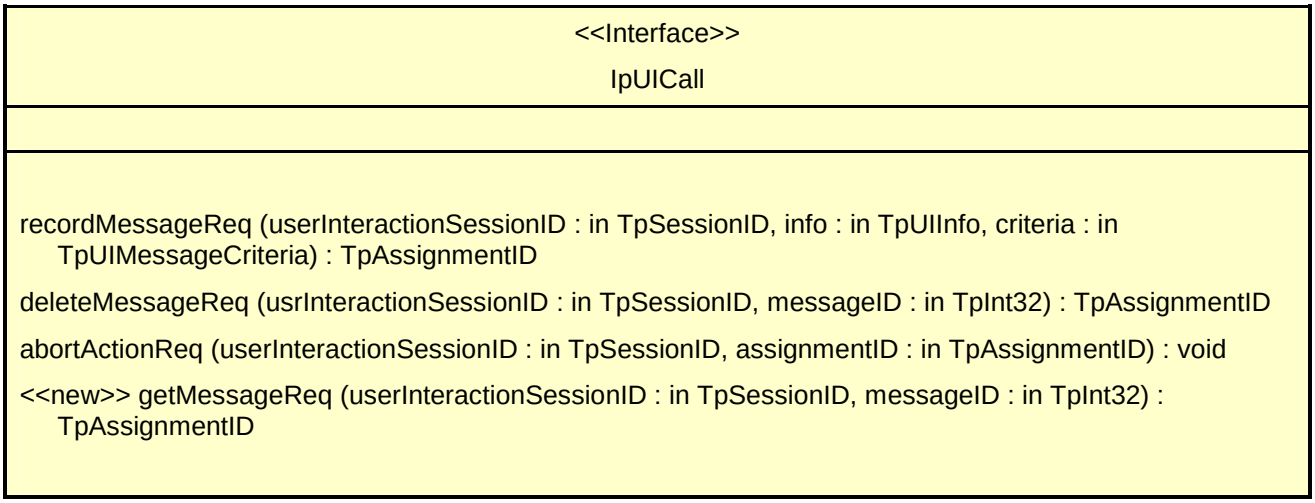
Specifies the fault that has been detected.

8.1.5 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.

This interface, or the IpUI interface, shall be implemented by a Generic User Interaction SCF as a minimum requirement. The minimum required methods of interface IpUI shall be implemented.



8.1.5.1 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

8.1.5.2 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

8.1.5.3 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

8.1.5.4 Method <<new>> getMessageReq()

This asynchronous method allows retrieving the recorded message content from the gateway. This method is applicable only to recorded messages.

Returns: assignmentID.

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

Parameters

userInteractionSessionID : 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_INVALID_NETWORK_STATE, P_ILLEGAL_ID, P_ID_NOT_FOUND

8.1.6 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 <<new>> getMessageRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, message : in TpUIInfo) : void <<new>> getMessageErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

8.1.6.1 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.

8.1.6.2 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.

8.1.6.3 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.

8.1.6.4 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.

8.1.6.5 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.

8.1.6.6 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.

8.1.6.7 Method <<new>> getMessageRes()

This method returns the message content if the message was retrieved successfully.

Parameters

userInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

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

message : in TpUIInfo

Specifies the UI Information containing the message content information.

8.1.6.8 Method <<new>> getMessageErr()

This method indicates that the request to retrieve 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 user interaction interface for a user interaction request.

error : in TpUIError

Specifies the error which led to the original request failing.

8.2 Generic User Interaction Administration Interface Classes

8.2.1 Interface Class IpUIAdminManager

Inherits from: IpService.

The Generic User Interaction Administration Manager Service interface is used by applications to manage user announcement and recorded messages on the gateway. This Service is represented by the IpUIAdminManager interface that interfaces to the service provided by the network. To handle responses and reports, the developer must implement IpAppUIAdminManager interface to provide the callback mechanism.

The application context will ensure that one application does not interfere with the messages of another application. The User Interaction Administration Manager Service Interface provides functions to manage the messages.

<<Interface>> IpUIAdminManager
<<new>> getMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID <<new>> putMessageReq (usrInteractionSessionID : in TpSessionID, info : in TpUIInfo) : TpAssignmentID <<new>> deleteMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID <<new>> getMessageListReq (usrInteractionSessionID : in TpSessionID, reset : in TpBoolean) : TpAssignmentID

8.2.1.1 Method <<new>> getMessageReq()

This asynchronous method allows retrieving the user announcement or recorded message content from the gateway.

Returns: assignmentID.

Specifies the ID assigned by the 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_INVALID_NETWORK_STATE, P_ILLEGAL_ID, P_ID_NOT_FOUND

8.2.1.2 Method <<new>> putMessageReq()

This asynchronous method allows putting a user announcement message content onto the gateway. The gateway will allocate the messageID and return it to the application on the putMessageRes() confirmation.

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.

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).

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_ILLEGAL_ID, P_ID_NOT_FOUND

8.2.1.3 Method <<new>> deleteMessageReq()

This asynchronous method allows deleting a user announcement or 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

8.2.1.4 Method <<new>> getMessageListReq()

This synchronous method allows the application to retrieve a list of Message Ids for all its recorded messages or user announcements.

Returns: assignmentID.

Specifies the ID assigned by the user interaction admin manager interface in order to correlate the response.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

reset : in TpBoolean

TRUE: Indicates that the application intends to obtain the list of messages starting from the beginning.

FALSE: Indicates that the application requests the next part of the list that has 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 the getMessageListRes(), 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

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

8.2.2 Interface Class IpAppUIAdminManager

Inherits from: IpInterface.

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

<<Interface>> IpAppUIAdminManager
<<new>> getMessageRes (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, message : in TpUIInfo) : void <<new>> getMessageErr (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void <<new>> deleteMessageRes (usrInteractionSessionID : in TpSessionID, response : in TpUIReport, assignmentID : in TpAssignmentID) : void <<new>> deleteMessageErr (usrInteractionSessionID : in TpSessionID, error : in TpUIError, assignmentID : in TpAssignmentID) : void <<new>> putMessageRes (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, messageID : in TpInt32) : void <<new>> putMessageErr (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void <<new>> getMessageListRes (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, messageIDList : in TpMessageIDList, final : in TpBoolean) : void <<new>> getMessageListErr (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

8.2.2.1 Method <<new>> getMessageRes()

This method returns the message content if the message was retrieved successfully.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

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

message : in TpUIInfo

Specifies the UI Information containing the message content information.

8.2.2.2 Method <<new>> getMessageErr()

This method indicates that the request to retrieve a message was not successful.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

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

error : in TpUIError

Specifies the error which led to the original request failing.

8.2.2.3 Method <<new>> deleteMessageRes()

This method indicates that the request to delete a message was successful.

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 user interaction interface for a user interaction request.

8.2.2.4 Method <<new>> deleteMessageErr()

This method indicates that the request to delete 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 user interaction interface for a user interaction request.

8.2.2.5 Method <<new>> putMessageRes()

This asynchronous method confirms that the request to put the message content was successful.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

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

messageID : in TpInt32

Specifies the message ID that was allocated by the gateway.

8.2.2.6 Method <<new>> putMessageErr()

This asynchronous method indicates that the request to put the message content resulted in an error.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

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

`error : in TpUIError`

Specifies the error which led to the original request failing.

8.2.2.7 Method <<new>> getMessageListRes()

This asynchronous method returns the result of a getMessageListReq() method. Whether there are still more messages that can be listed yet will be indicated with the final parameter.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction admin manager interface in order to correlate with the request.

`messageIDList : in TpMessageIDList`

Specifies the list of MessageIDs returned by the SCF.

`final : in TpBoolean`

Indication whether the returned list is the final part of the complete list (TRUE) or if there are still parts of the list to retrieve (FALSE).

8.2.2.8 Method <<new>> getMessageListErr()

This asynchronous method indicates that the request to list the messageIDs was not successful.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction admin manager interface in order to correlate with the request.

`error : in TpUIError`

Specifies the error which led to the original request failing.

9 State Transition Diagrams

9.1 Generic and Call User Interaction State Transition Diagrams

9.1.1 State Transition Diagrams for IpUIManager

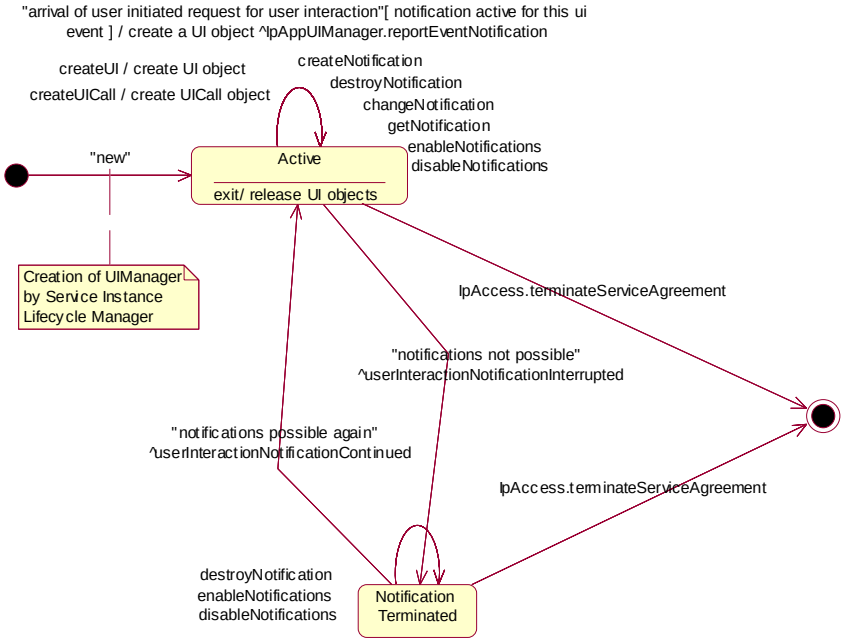


Figure 3: Application view on the UI Manager

9.1.1.1 Active State

In this state a relation between the Application and a User Interaction Service Capability Feature (Generic User Interaction or Call User Interaction) has been established. The application is now able to request creation of UI and/or UICall objects.

9.1.1.2 Notification Terminated State

When the UI manager is in the Notification terminated state, events requested with createNotification()/enableNotifications() will not be forwarded to the application. There can be multiple reasons for this: for instance it might be that the application receives more notifications than defined in the Service Level Agreement. Another example is that the SCS has detected it receives no notifications from the network due to e.g. a link failure. In this state no requests for new notifications will be accepted.

9.1.2 State Transition Diagrams for IpUI

The state transition diagram shows the application view on the User Interaction object.

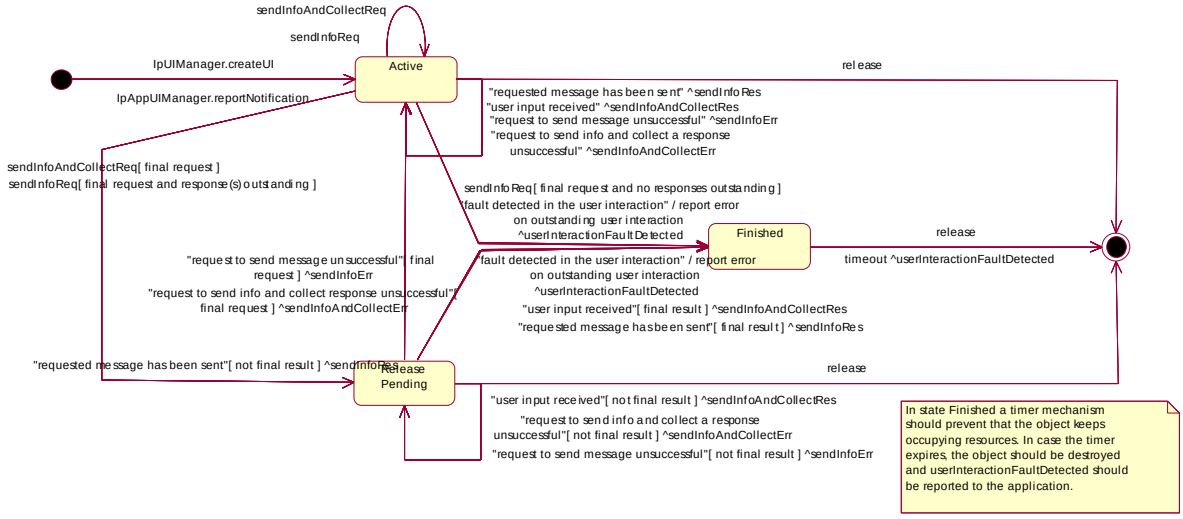


Figure 4: Application view on the UI object

9.1.2.1 Active State

In this state the UI object is available for requesting messages to be sent to the network.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

9.1.2.2 Release Pending State

A transition to this state is made when the Application has indicated that after a certain message no further messages need to be sent to the end-user. There are, however, still a number of messages that are not yet completed. When the last message is sent or when the last user interaction has been obtained, the UI object is destroyed.

In case the final request failed or the application requested to abort the final request, a transition is made back to the Active state.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

9.1.2.3 Finished State

In this state the user interaction has ended. The application can only release the UI object. Note that the application has to release the object itself as good Object Oriented practice requires that when an object is created on behalf of a certain entity, this entity is also responsible for destroying it when the object is no longer needed.

9.1.3 State Transition Diagrams for IpUICall

The state transition diagram shows the application view on the Call User Interaction object.

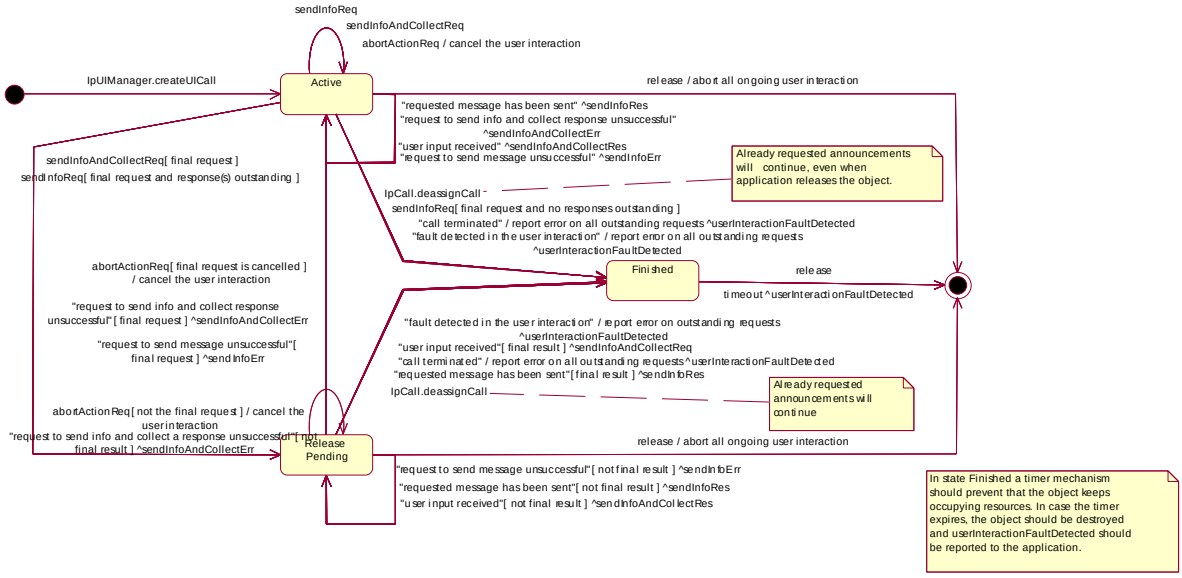


Figure 5: Application view on the UICall object

9.1.3.1 Active State

In this state a UICall object is available for announcements to be played to an end-user or obtaining information from the end-user.

When the application de-assigns the related Call or CallLeg object, a transition is made to the Finished state. However, all requested announcements will continue, even when the application releases the UICall object.

When the related call or call leg is terminated due to some reason, a transition is made to the Finished state, the operation userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

9.1.3.2 Release Pending State

A transition to this state is made when the Application has indicated that after a certain announcement no further announcements need to be played to the end-user. There are, however, still a number of announcements that are not yet completed. When the last announcement is played or when the last user interaction has been obtained, the UICall object is destroyed. In case the final request failed or the application requested to abort the final request, a transition is made back to the Active state.

When the application de-assigns the related Call or CallLeg object, the UICall object is destroyed. However, all requested announcements will continue.

When the related call or call leg is terminated due to some reason, a transition is made to the Finished state, the operation userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

9.1.3.3 Finished State

In this state the user interaction has ended. The application can only release the UICall object. Note that the application has to release the object itself as good Object Oriented practice requires that when an object is created on behalf of a certain entity, this entity is also responsible for destroying it when the object is no longer needed.

9.2 Generic User Interaction Administration State Transition Diagrams

9.2.1 State Transition Diagrams for IpUIAdminManager

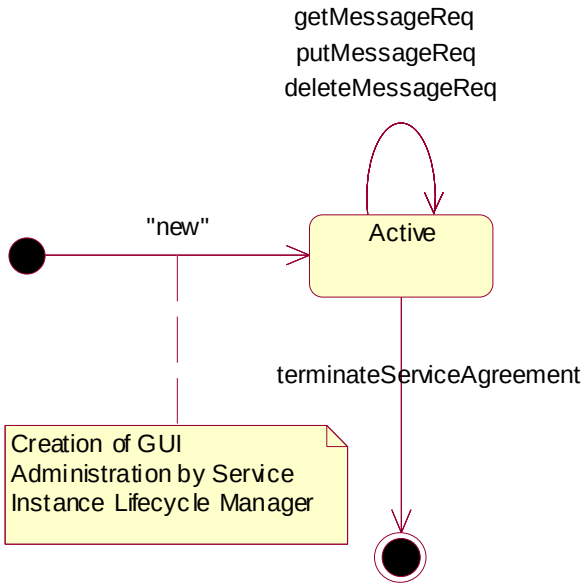


Figure 6: State Transition Diagram for User Interaction Administration

9.2.1.1 Active State

In this state, a relation between the Application and the Generic User Interaction Administration Service Capability Feature has been established. It allows the application to make specific requests of the service.

10 Service Properties

10.1 User Interaction Service Properties

The following table lists properties relevant for the User Interaction API.

Property	Type	Description
P_INFO_TYPE	INTEGER_SET	Specifies whether the UI SCS supports text or URLs etc. Allowed values are defined by TpUIInfoType.
P_SPEECH_RECOGNITION_SUPPORTED	BOOLEAN	Value: TRUE when the speech recognition features are supported.

The previous table lists properties related to capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description
P_SERVICE_CODE	INTEGER_SET	Specifies the service codes that may be used for notification requests.
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present.

11 Data Definitions

The following data type referenced in this clause is defined in ES 203 915-4-2:

`TpCallIdentifier`

The following data types referenced in this clause are defined in ES 203 915-4-3:

`TpMultiPartyCallIdentifier`
`TpCallLegIdentifier`

All other data types referenced but not defined in this clause are common data definitions which may be found in ES 203 915-2.

11.1 TpUIFault

Defines the cause of the UI fault detected.

Name	Value	Description
P_UI_FAULT_UNDEFINED	0	Undefined.
P_UI_CALL_ENDED	1	The related Call object has been terminated. Therefore, the UICall object is also terminated. No further interaction is possible with this object.

11.2 IpUI

Defines the address of an `IpUI` Interface.

11.3 IpUIRef

Defines a `Reference` to type `IpUI`.

11.4 IpAppUI

Defines the address of an `IpAppUI` Interface.

11.5 IpAppUIRef

Defines a `Reference` to type `IpAppUI`.

11.6 IpAppUIManager

Defines the address of an `IpAppUIManager` Interface.

11.7 IpAppUIManagerRef

Defines a `Reference` to type `IpAppUIManager`.

11.8 TpUICallIdentifier

Defines the `Sequence of Data Elements` that unambiguously specify the UICall object.

Structure Element Name	Structure Element Type	Structure Element Description
UICallRef	IpUICallRef	This element specifies the interface reference for the UICall object.
UserInteractionSessionID	TpSessionID	This element specifies the User Interaction session ID.

11.9 TpUICollectCriteria

Defines the `Sequence of Data Elements` that specify the additional properties for the collection of information, such as the end character, first character timeout, inter-character timeout, and maximum interaction time. The CollectMode element defines the type of data that is to be collected. DTMF and Voice Recognition can be used separately or in combination. The P_SPEECH_RECOGNITION_SUPPORTED property defines whether the voice recognition features are supported.

Structure Element Name	Structure Element Type
MinLength	TpInt32
MaxLength	TpInt32
EndSequence	TpString
StartTimeout	TpDuration
InterCharTimeout	TpDuration
CollectMode	TpUICollectMode
RecognitionCriteria	TpUIRecognitionCriteria

The structure elements specify the following criteria:

`MinLength:` Defines the minimum number of characters (e.g. digits) to collect. Applies to DTMF collection and voice recognition.

`MaxLength:` Defines the maximum number of characters (e.g. digits) to collect. Applies to DTMF collection and voice recognition.

`EndSequence:` Defines the character or characters which terminate an input of variable length, e.g. phone numbers. Applies to DTMF collection only.

`StartTimeout:` Specifies the value for the first character time-out timer. The timer is started when the announcement has been completed or has been interrupted. The user should enter the start of the response (e.g. first digit) before the timer expires. If the start of the response is not entered before the timer expires, the input is regarded to be erroneous. After receipt of the start of the response, which may be valid or invalid, the timer is stopped. Applies to DTMF collection and voice recognition.

`InterCharTimeOut:` Specifies the value for the inter-character time-out timer. The timer is started when a response (e.g. digit) is received, and is reset and restarted when a subsequent response is received. The responses may be valid or invalid. the announcement has been completed or has been interrupted. Applies to DTMF collection only.

`CollectMode:` Defines the type of collection to do. Applies to DTMF collection and voice recognition. The default is DTMF collection only.

`RecognitionCriteria:` Defines the criteria for voice recognition.

Input is considered successful if the following applies:

If the `EndSequence` is not present (i.e. an empty string):

- when the `InterCharTimeOut` timer expires; or

- when the number of valid digits received equals the `MaxLength`.

If the `EndSequence` is present:

- when the `InterCharTimeOut` timer expires; or
- when the `EndSequence` is received; or
- when the number of valid digits received equals the `MaxLength`.

In the case the number of valid characters received is less than the `MinLength` when the `InterCharTimeOut` timer expires or when the `EndSequence` is received, the input is considered erroneous.

The collected characters (including the `EndSequence`) are sent to the client application when input has been successful.

11.10 TpUIError

Defines the UI error codes.

Name	Value	Description
P_UI_ERROR_UNDEFINED	0	Undefined error.
P_UI_ERROR_ILLEGAL_INFO	1	The specified information (InfoId, InfoData, or InfoAddress) is invalid.
P_UI_ERROR_ID_NOT_FOUND	2	A legal InfoId is not known to the User Interaction service.
P_UI_ERROR_RESOURCE_UNAVAILABLE	3	The information resources used by the User Interaction service are unavailable, e.g. due to an overload situation.
P_UI_ERROR_ILLEGAL_RANGE	4	The values for minimum and maximum collection length are out of range.
P_UI_ERROR_IMPROPER_USER_RESPONSE	5	Improper user response.
P_UI_ERROR_ABANDON	6	The specified leg is disconnected before the send information completed.
P_UI_ERROR_NO_OPERATION_ACTIVE	7	There is no active User Interaction for the specified leg. Either the application did not start any User Interaction or the User Interaction was already finished when the abortActionReq() was called.
P_UI_ERROR_NO_SPACE_AVAILABLE	8	There is no more storage capacity to record the message when the recordMessageReq() operation was called.
P_UI_ERROR_RESOURCE_TIMEOUT	9	The request has been accepted by the resource but it did not report a result.

The call User Interaction object will be automatically de-assigned if the error P_UI_ERROR_ABANDON is reported, as a corresponding call or call leg object no longer exists.

11.11 TpUIEventCriteria

Defines the `Sequence of Data Elements` that specify the additional criteria for receiving a UI notification.

Structure Element Name	Structure Element Type	Description
OriginatingAddress	TpAddressRange	Defines the originating address for which the notification is requested.
DestinationAddress	TpAddressRange	Defines the destination address or address range for which the notification is requested.
ServiceCode	TpString	Defines a 2-digit code indicating the UI to be triggered. The value is operator specific.

11.12 TpUIEventCriteriaResultSet

Defines a set of TpUIEventCriteriaResult.

11.13 TpUIEventCriteriaResult

Defines a `sequence of data elements` that specify a requested event notification criteria with the associated assignmentID.

Structure Element Name	Structure Element Type	Structure Element Description
EventCriteria	TpUIEventCriteria	The event criteria that were specified by the application.
AssignmentID	TpInt32	The associated assignmentID. This can be used to disable the notification.

11.14 TpUIEventInfoDataType

Defines the type of the dataString parameter in the method userInteractionEventNotify.

Name	Value	Description
P_UI_EVENT_DATA_TYPE_UNDEFINED	0	Undefined (e.g. binary data)
P_UI_EVENT_DATA_TYPE_UNSPECIFIED	1	Unspecified data
P_UI_EVENT_DATA_TYPE_TEXT	2	Text
P_UI_EVENT_DATA_TYPE_USSD_DATA	3	USSD data starting with coding scheme

11.15 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.

11.16 TpUIIdentifierSet

Defines a `Numbered Set of Data Elements` of TpUIIdentifier.

11.17 TpUIInfo

Defines the `Tagged Choice of Data Elements` that specify the information to send to the user.

	Tag Element Type	
	TpUIInfoType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_INFO_ID	TpInt32	InfoId
P_UI_INFO_DATA	TpString	InfoData
P_UI_INFO_ADDRESS	TpURL	InfoAddress
P_UI_INFO_BIN_DATA	TpOctetSet	InfoBinData
P_UI_INFO_UUENCODED	TpString	InfoUUEncData
P_UI_INFO_MIME	TpOctetSet	InfoMimeData
P_UI_INFO_WAVE	TpOctetSet	InfoWaveData
P_UI_INFO_AU	TpOctetSet	InfoAuData
P_UI_INFO_VXML	TpString	InfoVXMLData
P_UI_INFO_SYNTHESIS	TpUISynthesisInfoData	InfoSynthData

The choice elements represent the following:

`InfoID:` defines the ID of the user information script or stream to send to an end-user. The values of this data type are operator specific.

`InfoData:` defines the data to be sent to an end-user's terminal. The data is free-format and the encoding is depending on the resources being used.

`InfoAddress:` defines the URL of the text, voice application script or stream to be either sent to an end-user's terminal or invoked in the network in order to carry out the interaction dialogue.

`InfoBinData:` defines the binary data to be sent to an end-user's terminal. The data is a free-format, 8-bit quantity that is guaranteed not to undergo any conversion when transmitted.

`InfoUUEncData:` defines the UUEncoded data to be sent to an end-user's terminal.

`InfoMimeData:` defines the MIME data to be sent to an end-user's terminal.

`InfoWaveData:` defines the WAVE data to be sent to an end-user's terminal.

`InfoAuData:` defines the AU data to be sent to an end-user's terminal.

`InfoVXMLData:` defines the TpString that describes the VXML (Voice XML) page that is sent to the server for execution and interaction with the end-user.
See http://www.w3.org/TR/2000/NOTE-voicexml-20000505/ for more information.

`InfoSynthData:` defines the TpUISynthesisInfoData that describes the content and how the speech synthesis will be done.

InfoSynthData allows the application to utilize the fundamental speech synthesis capabilities of the server without dependency VXML, while InfoVXMLData allows the application to send a complex VXML program (including call control, flow control, dynamic content, menuing, etc) to the server for execution with little change to the OSA application itself.

11.18 TpUIInfoType

Defines the type of the information to be sent to the user.

Name	Value	Description
P_UI_INFO_ID	0	The information to be sent to an end-user consists of an ID.
P_UI_INFO_DATA	1	The information to be sent to an end-user consists of a data string.
P_UI_INFO_ADDRESS	2	The information to be sent to an end-user consists of a URL.
P_UI_INFO_BIN_DATA	3	The information to be sent to an end-user consists of an 8 bit binary data set.
P_UI_INFO_UUENCODED	4	The information to be sent to an end-user consists of UUEncoded data.
P_UI_INFO_MIME	5	The information to be sent to the end-user consists of MIME encoded data.
P_UI_INFO_WAVE	6	The information to be sent to the end-user is .wav waveform data.
P_UI_INFO_AU	7	The information to be sent to the end-user is .au audio data.
P_UI_INFO_VXML	8	The information to be sent to the end-user is controlled by this VXML.
P_UI_INFO_SYNTHESIS	9	The information to be sent to an end-user is synthesized from text.

11.19 TpUIMessageCriteria

Defines the `Sequence of Data Elements` that specify the additional properties for the recording of a message.

Structure Element Name	Structure Element Type
EndSequence	TpString
MaxMessageTime	TpDuration
MaxMessageSize	TpInt32

The structure elements specify the following criteria:

`EndSequence:` Defines the character or characters which terminate an input of variable length, e.g. phone numbers.

`MaxMessageTime:` specifies the maximum duration in seconds of the message that is to be recorded.

`MaxMessageSize:` If this parameter is non-zero, it specifies the maximum size in bytes of the message that is to be recorded.

11.20 TpUIReport

Defines the UI reports if a response was requested.

Name	Value	Description
P_UI_REPORT_UNDEFINED	0	Undefined report.
P_UI_REPORT_INFO_SENT	1	Confirmation that the information has been sent.
P_UI_REPORT_INFO_COLLECTED	2	Information collected., meeting the specified criteria.
P_UI_REPORT_NO_INPUT	3	No information collected. The user immediately entered the delimiter character. No valid information has been returned.
P_UI_REPORT_TIMEOUT	4	No information collected. The user did not input any response before the input timeout expired.
P_UI_REPORT_MESSAGE_STORED	5	A message has been stored successfully.
P_UI_REPORT_MESSAGE_NOT_STORED	6	The message has not been stored successfully.
P_UI_REPORT_MESSAGE_DELETED	7	A message has been deleted successfully.
P_UI_REPORT_MESSAGE_NOT_DELETED	8	A message has not been deleted successfully.

11.21 TpUIResponseRequest

Defines the situations for which a response is expected following the User Interaction.

Name	Value	Description
P_UI_RESPONSE_REQUIRED	1	The User Interaction Call shall send a response when the request has completed.
P_UI_LAST_ANNOUNCEMENT_IN_A_ROW	2	This is the final announcement within a sequence. It might, however, be that additional announcements will be requested at a later moment. The User Interaction Call service may release any used resources in the network. The UI object will not be released.
P_UI_FINAL_REQUEST	4	This is the final request. The UI object will be released after the information has been presented to the user.

This parameter represents a so-called bitmask, i.e. the values can be added to derive the final meaning.

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.

11.23 TpUITargetObject

Defines the `Tagged Choice of Data Elements` that specify the object to perform User Interaction on.

	Tag Element Type	
	TpUITargetObjectType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_TARGET_OBJECT_CALL	TpCallIdentifier	Call
P_UI_TARGET_OBJECT_MULTI_PARTY_CALL	TpMultiPartyCallIdentifier	MultiPartyCall
P_UI_TARGET_OBJECT_CALL_LEG	TpCallLegIdentifier	CallLeg

11.24 TpUIVariableInfo

Defines the `Tagged Choice of Data Elements` that specify the variable parts in the information to send to the user.

	Tag Element Type	
	TpUIVariablePartType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_VARIABLE_PART_INT	TpInt32	VariablePartInteger
P_UI_VARIABLE_PART_ADDRESS	TpString	VariablePartAddress
P_UI_VARIABLE_PART_TIME	TpTime	VariablePartTime
P_UI_VARIABLE_PART_DATE	TpDate	VariablePartDate
P_UI_VARIABLE_PART_PRICE	TpPrice	VariablePartPrice

11.25 TpUIVariableInfoSet

Defines a `Numbered Set of Data Elements` of TpUIVariableInfo.

11.26 TpUIVariablePartType

Defines the type of the variable parts in the information to send to the user.

Name	Value	Description
P_UI_VARIABLE_PART_INT	0	Variable part is of type integer
P_UI_VARIABLE_PART_ADDRESS	1	Variable part is of type address
P_UI_VARIABLE_PART_TIME	2	Variable part is of type time
P_UI_VARIABLE_PART_DATE	3	Variable part is of type date
P_UI_VARIABLE_PART_PRICE	4	Variable part is of type price

11.27 TpUIEventNotificationInfo

Defines the `Sequence of Data Elements` that specify a UI event notification.

Structure Element Name	Structure Element Type	Structure Element Description
OriginatingAddress	TpAddress	Defines the originating address.
DestinationAddress	TpAddress	Defines the destination address.
ServiceCode	TpString	Defines a 2-digit code indicating the UI to be triggered. The value is operator specific.
DataTypeIndication	TpUIEventInfoDataType	Identifies the type of contents in UIEventData
UIEventData	TpOctetSet	Freely defined data according to the network policy. e.g. 7 bit USSD encoded

11.28 TpUISynthesisInfoData

Defines the `Sequence of Data Elements` that specify the information to use in generating the desired effects on the generated voice. The speech synthesis parameters or processing tags will be interpreted as hints and may be ignored by the speech synthesis engine. Note that the language is specified on the sendInfoReq() and sendInfoAndCollectReq() method calls.

The TextData field may contain the following tags to affect the processing of the text. The <break> tag specifies a timing pause. The <emp> tag specifies an emphasis on a word or phrase.

```
<break size="milliseconds">      // specifies a timing pause in the
                                 // middle of the text in milliseconds
<emp>A word</emp>               // apply emphasis to a word or words
```

Structure Element Name	Structure Element Type	Structure Element Description
SpeakerGender	TpUISynthesisGender	Defines the gender of the speaker.
SpeakerAge	TpUISynthesisAge	Defines the age of the speaker.
SpeakerRate	TpUISynthesisRate	Defines the rate of the speaker.
SpeakerRange	TpUISynthesisRange	Defines the range of the speaker.
TextData	TpString	Defines the text to synthesize into speech.
WordOverrideSet	TpUIWordOverrideSet	Defines the pronunciation overrides.

11.29 TpUISynthesisGender

Defines the UI reports if a response was requested.

Name	Value	Description
P_UI_GENDER_MALE	0	Male
P_UI_GENDER_FEMALE	1	Female

11.30 TpUISynthesisAge

Defines the UI reports if a response was requested.

Name	Value	Description
P_UI_AGE_CHILD	0	Child voice
P_UI_AGE_YOUNG_ADULT	1	Young adults voice
P_UI_AGE_ADULT	2	Adult voice
P_UI_AGE_OLDER_ADULT	3	Older adult voice

11.31 TpUISynthesisRate

Defines the rate of the speech.

Name	Value	Description
P_UI_RATE_SLOW	0	Slow speech rate
P_UI_RATE_AVERAGE	1	Average speech rate
P_UI_RATE_FAST	2	Fast speech rate

11.32 TpUISynthesisRange

Defines the range or liveliness of the speech.

Name	Value	Description
P_UI_RANGE_CALMER	0	Very Calm or monotone speech
P_UI_RANGE_CALM	1	Moderately calm speech
P_UI_RANGE_AVERAGE	2	Average speech
P_UI_RANGE_EXCITED	3	Moderately excited or lively speech
P_UI_RANGE_MORE_EXCITED	4	Excited or lively speech

11.33 TpUIWordOverrideSet

Defines a `Numbered Set of Data Elements` of TpUIWordOverride.

11.34 TpUIWordOverride

Defines the `Sequence of Data Elements` that specify the information to use in overriding the default pronunciation of a word.

Structure Element Name	Structure Element Type	Structure Element Description
Spelling	TpString	Defines the spelling of the word override.
PronounceType	TpUIPronounceType	Defines the type of pronunciation syntax.
PronounceAs	TpString	Defines how the spelling field should be pronounced.

11.35 TpUIPronounceType

Defines the pronunciation type.

The International Phonetic Alphabet (IPA) representation can be used to specify pronunciations. For more information see:

http://www2.arts.gla.ac.uk/IPA/ipachart.html

http://www.unicode.org/charts/PDF/U0250.pdf

Also, simple sound-alike replacements can be used, such as 'I triple E' for IEEE.

Name	Value	Description
P_UI_PRONOUNCE_IPA	0	The IPA pronunciation type
P_UI_PRONOUNCE_SOUNDSLIKE	1	The simple sounds like replacement type

11.36 TpUICollectMode

Defines the type of collection.

Name	Value	Description
P_UI_COLLECT_MODE_DTMF	0	Collect DTMF digits only
P_UI_COLLECT_MODE_VOICE	1	Collect Voice recognized data
P_UI_COLLECT_MODE_DTMFANDVOICE	2	Collect both DTMF digits and voice recognized data

11.37 TpUIRecognitionCriteria

Defines the `Sequence of Data Elements` that specify the additional properties for the collection of information in the form of voice recognition according to the specified grammar.

Structure Element Name	Structure Element Type
SpeakerID	TpUIRecognitionSpeakerID
Properties	TpUIRecognitionPropertySet
Grammar	TpUIRecognitionGrammar

The structure elements specify the following criteria:

SpeakerID: Defines the user identifier associating a user with a speech profile known to the recognition engine, which provides a hint for better quality.

Properties: Defines the properties set for additional information to the speech recognition engine.

Grammar: Defines the syntax of the language to be recognized.

11.38 TpUIRecognitionSpeakerID

Defines a user identifier string that identifies the speaker and is a hint to whose voice is to be recognized.

11.39 TpUIRecognitionPropertySet

Defines a `Numbered Set of Data Elements` of TpUIRecognitionProperty.

11.40 TpUIRecognitionProperty

Defines the `Sequence of Data Elements` that specify the additional properties for the recognition engine. The TpUIRecognitionProperty is a hint to the recognition engine on how it should interpret the input.

Structure Element Name	Structure Element Type
PropertyName	TpString
PropertyValue	TpString

The structure elements specify the following criteria:

PropertyName: Defines the name of the property.

PropertyValue: Defines the value of the property.

The defined properties are:

P_RECOGNITION_PROPERTY_CONFIDENCE_LEVEL - The speech recognition confidence level, a float value in the range of 0,0 to 1,0. Results are rejected when the recognitions engine's confidence in its interpretation is below this threshold. A value of 0,0 means minimum confidence is needed for a recognition, and a value of 1,0 requires maximum confidence. The default value is 0,5.

P_RECOGNITION_PROPERTY_SENSITIVITY - Set the sensitivity level. A value of 1,0 means that it is highly sensitive to quiet input. A value of 0,0 means it is least sensitive to noise. The default value is 0,5.

P_RECOGNITION_PROPERTY_SPEEDVSACCURACY - A hint specifying the desired balance between speed vs. accuracy. A value of 0,0 means fastest recognition. A value of 1,0 means best accuracy. The default is value 0,5.

P_RECOGNITION_PROPERTY_COMPLETE_TIMEOUT - The speech timeout value to use when an active grammar is matched.

11.41 TpUIRecognitionGrammar

Defines a string that consists of an inline grammar that specifies the syntax of the speech to be recognized. The format of this string is based on a subset of the Voice XML 1.0 grammar element tag. The in-line grammar text must be enclosed within <grammar> ...</grammar> element tags. The contents of the grammar specifies the allowable input that the voice recognition will accept. The Voice XML grammar specifies the set of utterances that a user may speak to perform an action and specifies the corresponding string value for the result.

The following table describes the features that provide a language for describing context-free grammars.

Feature	Purpose
word or words	(terminals, tokens) need not be quoted
[x]	optional x
(...)	Grouping
x {value text}	arbitrary value text may be associated with x
x*	0 or more occurrences of x
x+	1 or more occurrences of x
x y z ...	a sequence of x then y then z then ...
x | y | z | ...	a set of alternatives of x or y or z or ...
<rule>	rule names (non-terminals) are enclosed in <>
<rule> = x;	a private rule definition
public <rule> = x;	a public rule definition

The format of the grammar tag is:

```
<grammar> grammar content </grammar>
```

The grammar defines a possible set of utterances. The text of the utterance itself is used as the value, if the value text is not explicitly specified with {value}.

This form is particularly convenient for expressing simple lists of alternative ways of saying the same thing, for example:

```
<grammar>
     [please] help [me] [please] | [please] I (need | want) help [please]
</grammar>

<grammar>
    hamburger | burger {hamburger} | (chicken [sandwich]) {chicken}
</grammar>
```

In the first example, any of the ways of saying "help" result in a valid response. In the second example, the user may say "hamburger" or "burger" and the response will be given the value "hamburger", or the user may say "chicken" or "chicken sandwich" and the result will be given the value "chicken".

If the grammar can not be matched, then a sendInfoAndCollectErr will result, with a P_IMPROPER_USER_RESPONSE.

For a better description and further examples of in-line grammar creation see:

"Speech Recognition Grammar Specification Version 1":
http://www.w3.org/TR/2004/REC-speech-grammar-20040316/

11.42 TpMessageIDList

This data type defines a `Numbered List of Data Elements` of type TpInt32.

12 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	Information id unknown.
P_ILLEGAL_RANGE	The values for minimum and maximum collection length are out of range.
P_INVALID_COLLECTION_CRITERIA	Invalid collection criteria specified.

Each exception class contains the following structure.

Structure Element Name	Structure Element Type	Structure Element Description
ExtraInformation	TpString	Carries extra information to help identify the source of the exception, e.g. a parameter name.

Annex A (normative): OMG IDL Description of User Interaction SCF

The OMG IDL representation of this interface specification is contained in text files (ui_data.idl and ui_interfaces.idl contained in archive es_20391505v010101p0.zip) which accompany the present document.

Annex B (informative): W3C WSDL Description of User Interaction SCF

The W3C WSDL representation of this interface specification is contained in text files (ui_interfaces.wsdl and ui_data.wsdl contained in archive es_20391505v010101p0.zip) which accompany the present document.

Annex C (informative): Java™ API Description of the User Interaction SCF

The Java™ API realisation of this interface specification is produced in accordance with the Java™ Realisation rules defined in ES 203 915-1. These rules aim to deliver for Java™, a developer API, provided as a realisation, supporting a Java™ API that represents the UML specifications. The rules support the production of both J2SE™ and J2EE™ versions of the API from the common UML specifications.

The J2SE™ representation of this interface specification is provided as Java™ Code, contained in archive 20391505J2SE.zip.

The J2EE™ representation of this interface specification is provided as Java™ Code, contained in archive 20391505J2EE.zip.

Both these archives can be found in es_20391505v010101p0.zip which accompanies the present document.

Annex D (informative): Contents of 3GPP OSA R6 User Interaction

All of the present document is relevant for TS 129 198-5 V6 (Release 6).

Annex E (informative): Description of Generic User Interaction SCF for 3GPP2 cdma2000 networks

This annex is intended to define the OSA API Stage 3 interface definitions and it provides the complete OSA specifications. It is an extension of OSA API specifications capabilities to enable operation in cdma2000 systems environment. They are in alignment with 3GPP2 Stage 1 requirements and Stage 2 architecture defined in [52], [53] and [54] of ES 203 915-1, clause 2. These requirements are expressed as additions to and/or exclusions from the 3GPP Release 6 specification. The information given here is to be used by developers in 3GPP2 cdma2000 network architecture to interpret the 3GPP OSA specifications.

E.1 General Exceptions

The terms 3GPP and UMTS are not applicable for the cdma2000 family of standards. Nevertheless these terms are used (TR 121 905) mostly in the broader sense of "3G Wireless System". If not stated otherwise there are no additions or exclusions required.

CAMEL and CAP mappings are not applicable for cdma2000 systems.

E.2 Specific Exceptions

E.2.1 Clause 1: Scope

There are no additions or exclusions.

E.2.2 Clause 2: References

Normative references on TS 123 078 and on TS 129 078 are not applicable for cdma2000 systems.

E.2.3 Clause 3: Definitions and abbreviations

There are no additions or exclusions.

E.2.4 Clause 4: Generic and Call User Interaction SCF

There are no additions or exclusions.

E.2.5 Clause 5: Class Diagrams

The sequence diagrams are included as implementation examples. Individual specifications for the sequences for Alarm Call, Call Barring 1, Network Controlled Notifications, Prepaid and Pre-Paid with Advice of Charge (AoC) have yet to be specified. In particular, there are specifications for Call Barring, Prepaid and Pre-Paid with AoC that are implemented in cdma2000 networks differently than shown in these sequence diagrams. These are for further study and are not part of this release. There are no additions or exclusions to this clause except the Caveat stated above.

E.2.6 Clause 6: Class Diagrams

There are no additions or exclusions.

E.2.7 Clause 7: The Service Interface Specifications

There are no additions or exclusions.

E.2.8 Clause 8: Generic User Interaction Interface Classes Definitions

There are no additions or exclusions.

E.2.9 Clause 9: State Transition Diagrams

There are no additions or exclusions.

E.2.10 Clause 10: Service Properties

There are no additions or exclusions.

E.2.11 Clause 11: Data Definitions

There are no additions or exclusions.

E.2.12 Clause 12: Exception Classes

There are no additions or exclusions.

E.2.13 Annex A (normative): OMG IDL Description of Generic User Interaction SCF

There are no additions or exclusions.

E.2.14 Annex B (informative): W3C WSDL Description of Generic User Interaction SCF

There are no additions or exclusions.

E.2.15 Annex C (informative): Java™ API Description of Generic User Interaction SCF

There are no additions or exclusions.

Annex F (informative): Record of changes

The following is a list of the changes made to the present document for each release. The list contains the names of all changed, deprecated, added or removed items in the specifications and not the actual changes. Any type of change information that is important to the reader is put in the final clause of this annex.

Changes are specified as changes to the prior major release, but every minor release will have its own part of the table allowing the reader to know when the actual change was made.

F.1 Interfaces

F.1.1 New

Identifier	Comments
Interfaces added in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
IpUIAdminManager	
IpAppUIAdminManager	

F.1.2 Deprecated

Identifier	Comments
Interfaces deprecated in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.1.3 Removed

Identifier	Comments
Interfaces removed in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.2 Methods

F.2.1 New

Identifier	Comments
Methods added in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
IpUICall.getMessageReq()	
IpAppUICall.getMessageRes()	
IpAppUICall.getMessage.Err()	
IpUIAdminManager.getMessageReq()	
IpUIAdminManager.putMessageReq()	
IpUIAdminManager.deleteMessageReq()	
IpAppUIAdminManager.getMessageRes()	
IpAppUIAdminManager.getMessageErr()	
IpAppUIAdminManager.putMessageRes()	
IpAppUIAdminManager.putMessageErr()	
IpAppUIAdminManager.deleteMessageRes()	
IpAppUIAdminManager.deleteMessageErr()	
IpUIAdminManager.getMessageListReq()	
IpAppUIAdminManager.getMessageListRes()	
IpAppUIAdminManager.getMessageListErr()	
IpAppUIManager.abortMultipleUserInteractions()	

F.2.2 Deprecated

Identifier	Comments
Methods deprecated in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.2.3 Modified

Identifier	Comments
Methods modified in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.2.4 Removed

Identifier	Comments
Methods removed in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
IpAppUIManager.reportNotification()	Broken, already deprecated

F.3 Data Definitions

F.3.1 New

Identifier	Comments
Data Definitions added in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
TpUISynthesisInfoData	
TpUISynthesisGender	
TpUISynthesisAge	
TpUISynthesisRate	
TpUISynthesisRange	
TpUIWordOverrideList	
TpUIWordOverride	
TpUIPronounceType	
TpUICollectMode	
TpUIRecognitionCriteria	
TpUIRecognitionSpeakerID	
TpUIRecognitionPropertyList	
TpUIRecognitionProperty	
TpUIRecognitionGrammar	
TpMessageIDList	
TpUIIdentifierSet	

F.3.2 Modified

Identifier	Comments
Data Definitions modified in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
TpUICollectCriteria	Fields CollectMode and RecognitionCriteria added
TpUIInfo	Fields InfoVXMLData and InfoSynthData added
TpUIInfoType	Fields P_UI_INFO_VXML and P_UI_INFO_SYNTHESIS added
TpUIWordOverrideList	Renamed to TpUIWordOverrideSet
TpUISynthesisInfoData	Field name WordOverrideList changed to WordOverrideSet, type of field TpUIWordOverrideList changed to TpUIWordOverrideSet
TpUIRecognitionPropertyList	Renamed to TpUIRecognitionPropertySet
TpUIRecognitionCriteria	Type of Properties field changed to TpUIRecognitionPropertySet

F.3.3 Removed

Identifier	Comments
Data Definitions removed in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
TpUIEventInfo	Unused data type

F.4 Service Properties

F.4.1 New

Identifier	Comments
Service Properties added in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
P_SPEECH_RECOGNITION_SUPPORTED	
P_NOTIFICATION_ADDRESS_RANGES	Replaces P_TRIGGERING_ADDRESSES

F.4.2 Deprecated

Identifier	Comments
Service Properties deprecated in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.4.3 Modified

Identifier	Comments
Service Properties modified in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
P_INFO_TYPE	Range of values now defined by TpUIInfoType.

F.4.4 Removed

Identifier	Comments
Service Properties removed in ES 203 915-5 version 1.1.1 (Parlay 5.0)	
P_TRIGGERING_ADDRESSES	Replaced by P_NOTIFICATION_ADDRESS_RANGES

F.5 Exceptions

F.5.1 New

Identifier	Comments
Exceptions added in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.5.2 Modified

Identifier	Comments
Exceptions modified in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.5.3 Removed

Identifier	Comments
Exceptions removed in ES 203 915-5 version 1.1.1 (Parlay 5.0)	

F.6 Others

None.

History

Document history		
V1.1.1	February 2005	Membership Approval Procedure MV 20050408: 2005-02-08 to 2005-04-08
V1.1.1	April 2005	Publication