

# ETSI TS 129 198-5 V5.4.0 (2003-09)

---

*Technical Specification*

**Universal Mobile Telecommunications System (UMTS);  
Open Service Access (OSA);  
Application Programming Interface (API);  
Part 5: Generic user interaction  
(3GPP TS 29.198-05 version 5.4.0 Release 5)**

---



---

Reference

RTS/TSGN-0529198-05v540

---

Keywords

UMTS

**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, send your comment to:

[editor@etsi.org](mailto:editor@etsi.org)

---

**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 2003.  
All rights reserved.

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

---

## 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 Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under <http://webapp.etsi.org/key/queryform.asp>.

# Contents

Intellectual Property Rights .....	2
Foreword.....	2
Foreword.....	6
Introduction .....	6
1 Scope .....	8
2 References .....	8
3 Definitions and abbreviations.....	9
3.1 Definitions .....	9
3.2 Abbreviations .....	9
4 Generic and Call User Interaction SCF.....	9
4.1 General requirements on support of methods.....	10
5 Sequence Diagrams .....	10
5.1 Alarm Call .....	10
5.2 Call Barring 1 .....	12
5.3 Network Controlled Notifications .....	13
5.4 Prepaid.....	15
5.5 Pre-Paid with Advice of Charge (AoC).....	17
6 Class Diagrams.....	20
7 The Service Interface Specifications .....	21
7.1 Interface Specification Format .....	21
7.1.1 Interface Class .....	21
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 .....	22
7.3.1 Overview .....	22
7.4 Generic Service Interface .....	22
7.4.1 Interface Class IpService .....	22
7.4.1.1 Method setCallback().....	23
7.4.1.2 Method setCallbackWithSessionID().....	23
8 Generic User Interaction Interface Classes .....	23
8.1 Interface Class IpUIManager .....	24
8.1.1 Method createUI().....	24
8.1.2 Method createUICall().....	25
8.1.3 Method createNotification().....	25
8.1.4 Method destroyNotification() .....	26
8.1.5 Method changeNotification().....	26
8.1.6 Method getNotification() .....	26
8.1.7 Method <<new>> enableNotifications().....	27
8.1.8 Method <<new>> disableNotifications().....	27
8.2 Interface Class IpAppUIManager.....	28
8.2.1 Method userInteractionAborted() .....	28
8.2.2 Method <<deprecated>> reportNotification() .....	28
8.2.3 Method userInteractionNotificationInterrupted().....	29
8.2.4 Method userInteractionNotificationContinued().....	29
8.2.5 Method <<new>> reportEventNotification().....	29
8.3 Interface Class IpUI.....	30
8.3.1 Method sendInfoReq().....	30

8.3.2	Method sendInfoAndCollectReq()	31
8.3.3	Method release()	32
8.3.4	Method <<new>> setOriginatingAddress()	32
8.3.5	Method <<new>> getOriginatingAddress()	33
8.4	Interface Class IpAppUI	33
8.4.1	Method sendInfoRes()	34
8.4.2	Method sendInfoErr()	34
8.4.3	Method sendInfoAndCollectRes()	35
8.4.4	Method sendInfoAndCollectErr()	35
8.4.5	Method userInteractionFaultDetected()	35
8.5	Interface Class IpUICall	35
8.5.1	Method recordMessageReq()	36
8.5.2	Method deleteMessageReq()	36
8.5.3	Method abortActionReq()	37
8.6	Interface Class IpAppUICall	37
8.6.1	Method recordMessageRes()	38
8.6.2	Method recordMessageErr()	38
8.6.3	Method deleteMessageRes()	39
8.6.4	Method deleteMessageErr()	39
8.6.5	Method abortActionRes()	39
8.6.6	Method abortActionErr()	40
9	State Transition Diagrams	40
9.1	State Transition Diagrams for IpUIManager	40
9.1.1	Active State	41
9.1.2	Notification Terminated State	41
9.2	State Transition Diagrams for IpUI	41
9.2.1	Active State	41
9.2.2	Release Pending State	41
9.2.3	Finished State	42
9.3	State Transition Diagrams for IpUICall	42
9.3.1	Active State	42
9.3.2	Release Pending State	42
9.3.3	Finished State	43
10	Service Properties	43
10.1	User Interaction Service Properties	43
11	Data Definitions	43
11.1	TpUIFault	44
11.2	IpUI	44
11.3	IpUIRef	44
11.4	IpAppUI	44
11.5	IpAppUIRef	44
11.6	IpAppUIManager	44
11.7	IpAppUIManagerRef	44
11.8	TpUICallIdentifier	44
11.9	TpUICollectCriteria	45
11.10	TpUIError	46
11.11	TpUIEventCriteria	46
11.12	TpUIEventCriteriaResultSet	46
11.13	TpUIEventCriteriaResult	46
11.14	TpUIEventInfo	47
11.15	TpUIEventInfoDataType	47
11.16	TpUIIdentifier	47
11.17	TpUIInfo	47
11.18	TpUIInfoType	48
11.19	TpUIMessageCriteria	48
11.20	TpUIReport	49
11.21	TpUIResponseRequest	49
11.22	TpUITargetObjectType	49
11.23	TpUITargetObject	50
11.24	TpUIVariableInfo	50

11.25	TpUIVariableInfoSet.....	50
11.26	TpUIVariablePartType.....	50
11.27	TpUIEventNotificationInfo.....	51
12	Exception Classes.....	51
<b>Annex A (normative):</b>	<b>OMG IDL Description of User Interaction SCF.....</b>	<b>52</b>
<b>Annex B (informative):</b>	<b>WSDL Description of User Interaction SCF.....</b>	<b>53</b>
<b>Annex C (informative):</b>	<b>Java API Description of the User Interaction SCF.....</b>	<b>54</b>
<b>Annex D (informative):</b>	<b>Change history.....</b>	<b>55</b>
History.....		56

---

## Foreword

This Technical Specification has been produced by the 3<sup>rd</sup> Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

---

## Introduction

The present document is part 5 of a multi-part TS covering the 3<sup>rd</sup> Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API), as identified below. The **API specification** (3GPP TS 29.198) is structured in the following Parts:

Part 1:	"Overview";	
Part 2:	"Common Data Definitions";	
Part 3:	"Framework";	
Part 4:	"Call Control";	
	Sub-part 1: "Call Control Common Definitions";	(new in 3GPP Release 5)
	Sub-part 2: "Generic Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 3: "Multi-Party Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 4: "Multi-Media Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 5: "Conference Call Control SCF";	(not part of 3GPP Release 5)
<b>Part 5:</b>	<b>"User Interaction SCF";</b>	
Part 6:	"Mobility SCF";	
Part 7:	"Terminal Capabilities SCF";	
Part 8:	"Data Session Control SCF";	
Part 9:	"Generic Messaging SCF";	(not part of 3GPP Release 5)
Part 10:	"Connectivity Manager SCF";	(not part of 3GPP Release 5)
Part 11:	"Account Management SCF";	
Part 12:	"Charging SCF".	
Part 13:	"Policy Management SCF";	(new in 3GPP Release 5)
Part 14:	"Presence and Availability Management SCF";	(new in 3GPP Release 5)

The **Mapping specification of the OSA APIs and network protocols** (3GPP TR 29.998) is also structured as above. A mapping to network protocols is however not applicable for all Parts, but the numbering of Parts is kept. Also in case a Part is not supported in a Release, the numbering of the parts is maintained.

Table: Overview of the OSA APIs &amp; Protocol Mappings 29.198 &amp; 29.998-family

OSA API specifications 29.198-family					OSA API Mapping - 29.998-family	
29.198-01	Overview				29.998-01	Overview
29.198-02	Common Data Definitions				29.998-02	<i>Not Applicable</i>
29.198-03	Framework				29.998-03	<i>Not Applicable</i>
Call Control (CC) SCF	29.198-04-1	29.198-04-2	29.198-04-3	29.198-04-4	29.998-04-1	Generic Call Control – CAP mapping
	Common CC data definitions	Generic CC SCF	Multi-Party CC SCF	Multi-media CC SCF	29.998-04-2	<i>Generic Call Control – INAP mapping</i>
					29.998-04-3	<i>Generic Call Control – Megaco mapping</i>
					29.998-04-4	Multiparty Call Control – SIP mapping
<b>29.198-05</b>	<b>User Interaction SCF</b>				29.998-05-1	User Interaction – CAP mapping
					29.998-05-2	<i>User Interaction – INAP mapping</i>
					29.998-05-3	<i>User Interaction – Megaco mapping</i>
					29.998-05-4	User Interaction – SMS mapping
29.198-06	Mobility SCF				29.998-06	User Status and User Location – MAP mapping
29.198-07	Terminal Capabilities SCF				29.998-07	<i>Not Applicable</i>
29.198-08	Data Session Control SCF				29.998-08	Data Session Control – CAP mapping
29.198-09	<i>Generic Messaging SCF</i>				29.998-09	<i>Not Applicable</i>
29.198-10	<i>Connectivity Manager SCF</i>				29.998-10	<i>Not Applicable</i>
29.198-11	Account Management SCF				29.998-11	<i>Not Applicable</i>
29.198-12	Charging SCF				29.998-12	<i>Not Applicable</i>
29.198-13	Policy Management SCF				29.998-13	<i>Not Applicable</i>
29.198-14	Presence & Availability Management SCF				29.998-14	<i>Not Applicable</i>

---

# 1 Scope

This 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

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).

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI SPAN 12 and the Parlay Consortium, in co-operation with a number of JAIN™ Community member companies.

---

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 29.198-1: "Open Service Access; Application Programming Interface; Part 1: Overview".

[2] 3GPP TS 22.127: "Stage 1 Service Requirement for the Open Service Access (OSA) (Release 5)".

[3] 3GPP TS 23.127: "Virtual Home Environment (Release 5)".

---

## 3 Definitions and abbreviations

### 3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.198-1 [1] apply.

### 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.198-1 [1] apply.

---

## 4 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.

The following table 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
reportNotification	sendInfoAndCollectRes
userInteractionAborted	sendInfoAndCollectErr
userInteractionNotificationInterrupted	release
userInteractionNotificationContinued	userInteractionFaultDetected
changeNotification	
getNotification	
enableNotifications	
disableNotifications	

The following table 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

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.

The following clauses describe each aspect of the Generic User Interaction Service Capability Feature (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 show 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 show 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 of this specification.

## 4.1 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.

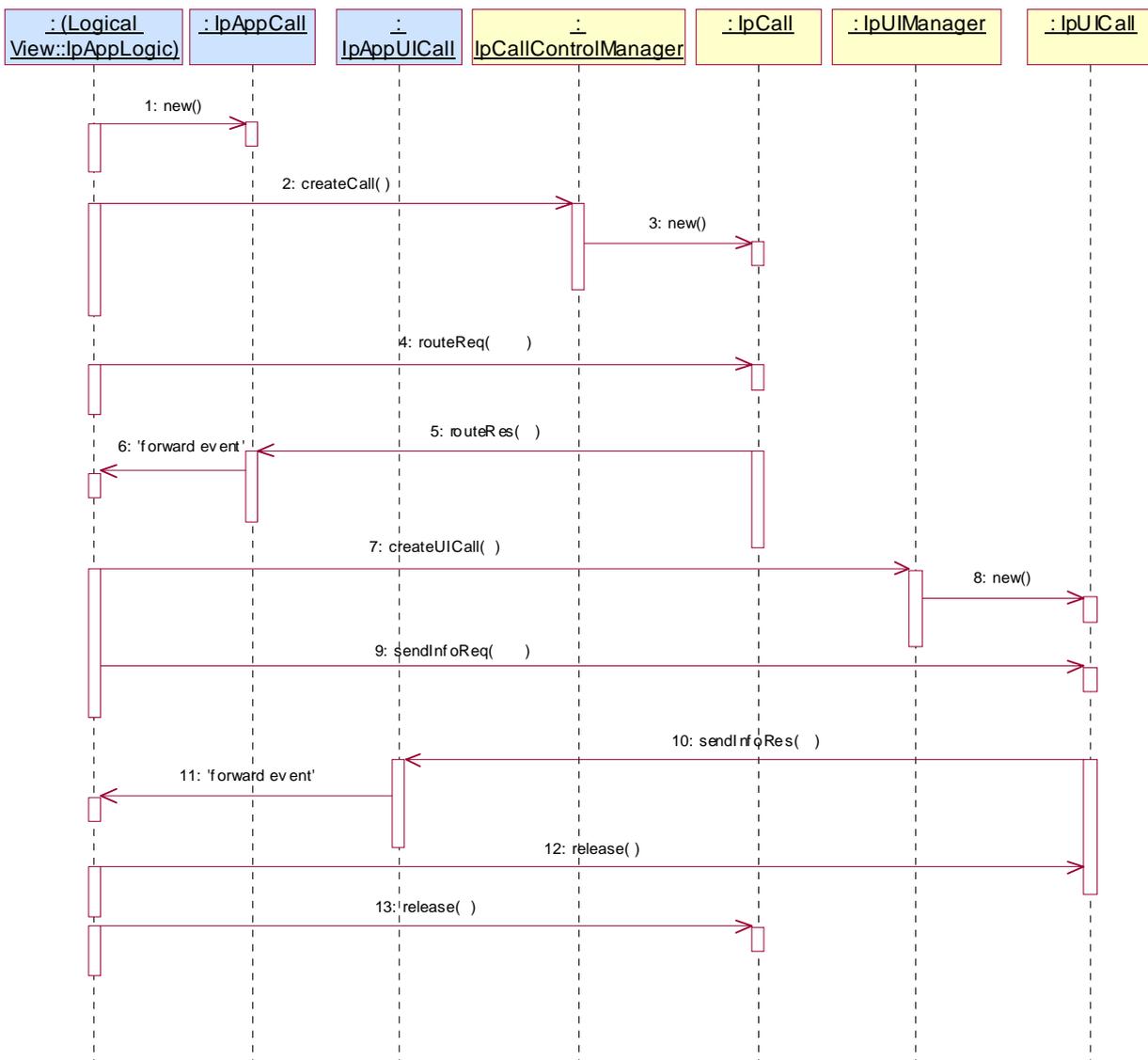
Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

---

# 5 Sequence Diagrams

## 5.1 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.

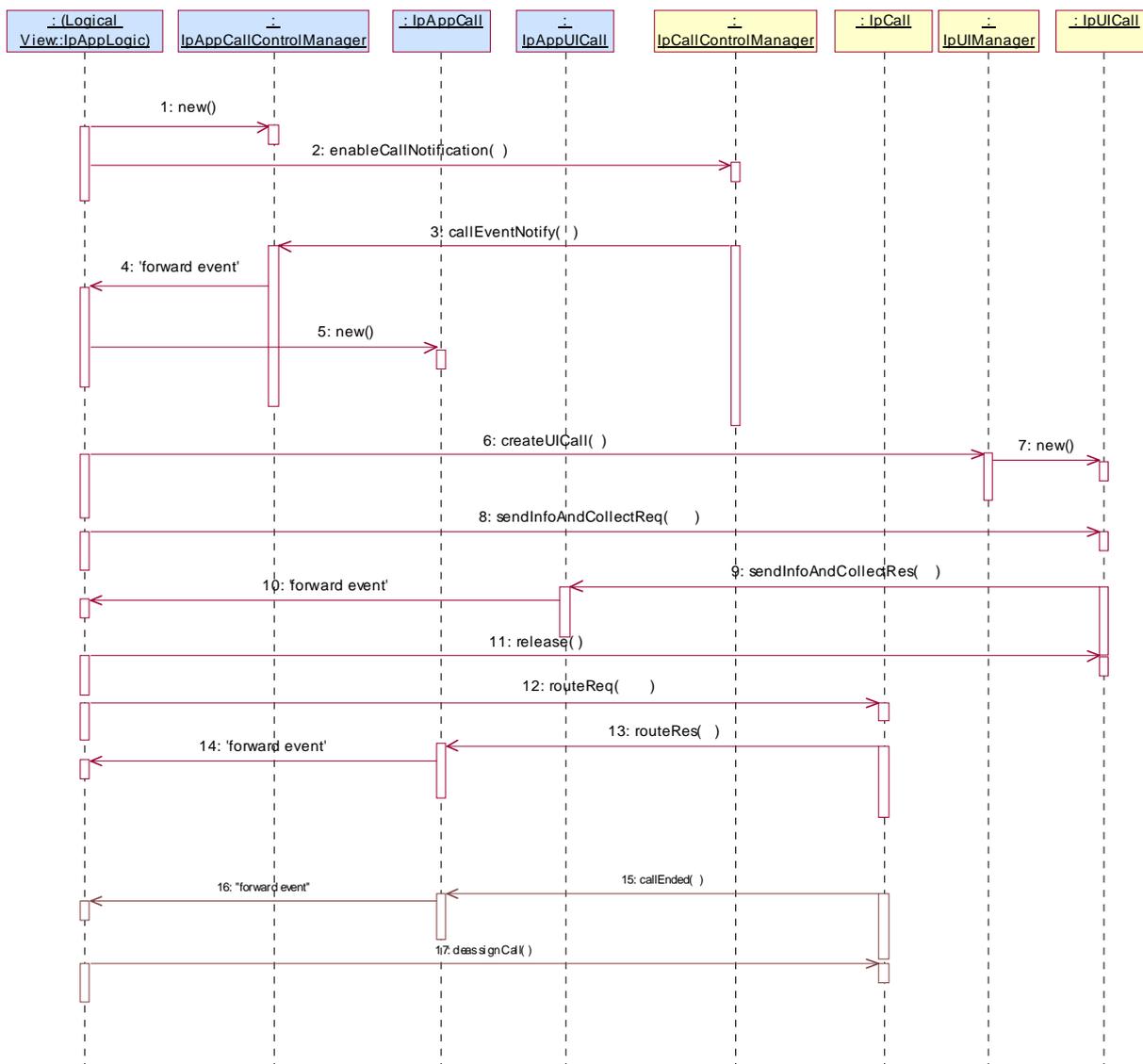
11: The event is forwarded to the application logic.

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.

## 5.2 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.

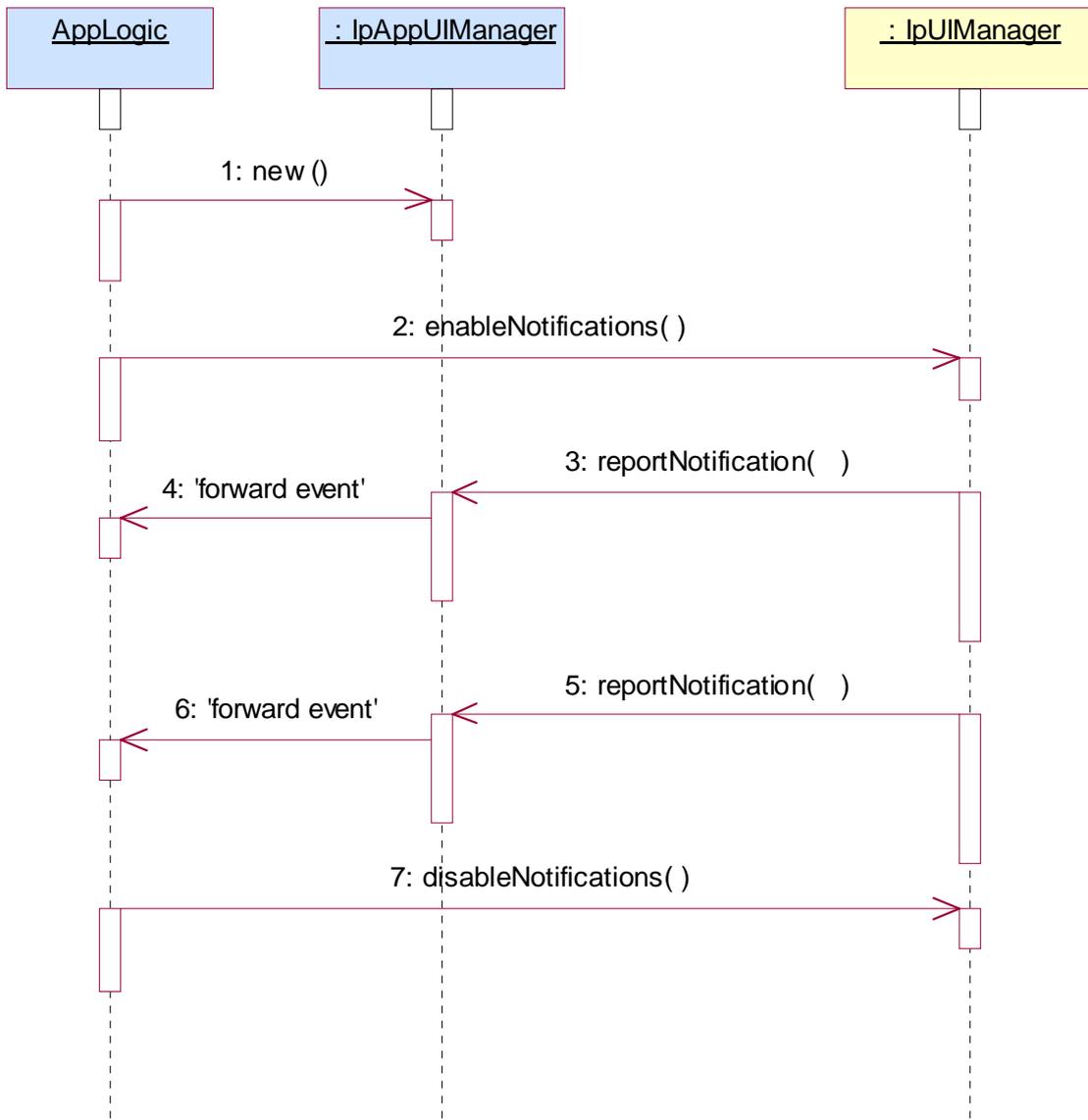
2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range prompted for a password before the call is allowed to progress. When 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.

## 5.3 Network Controlled Notifications

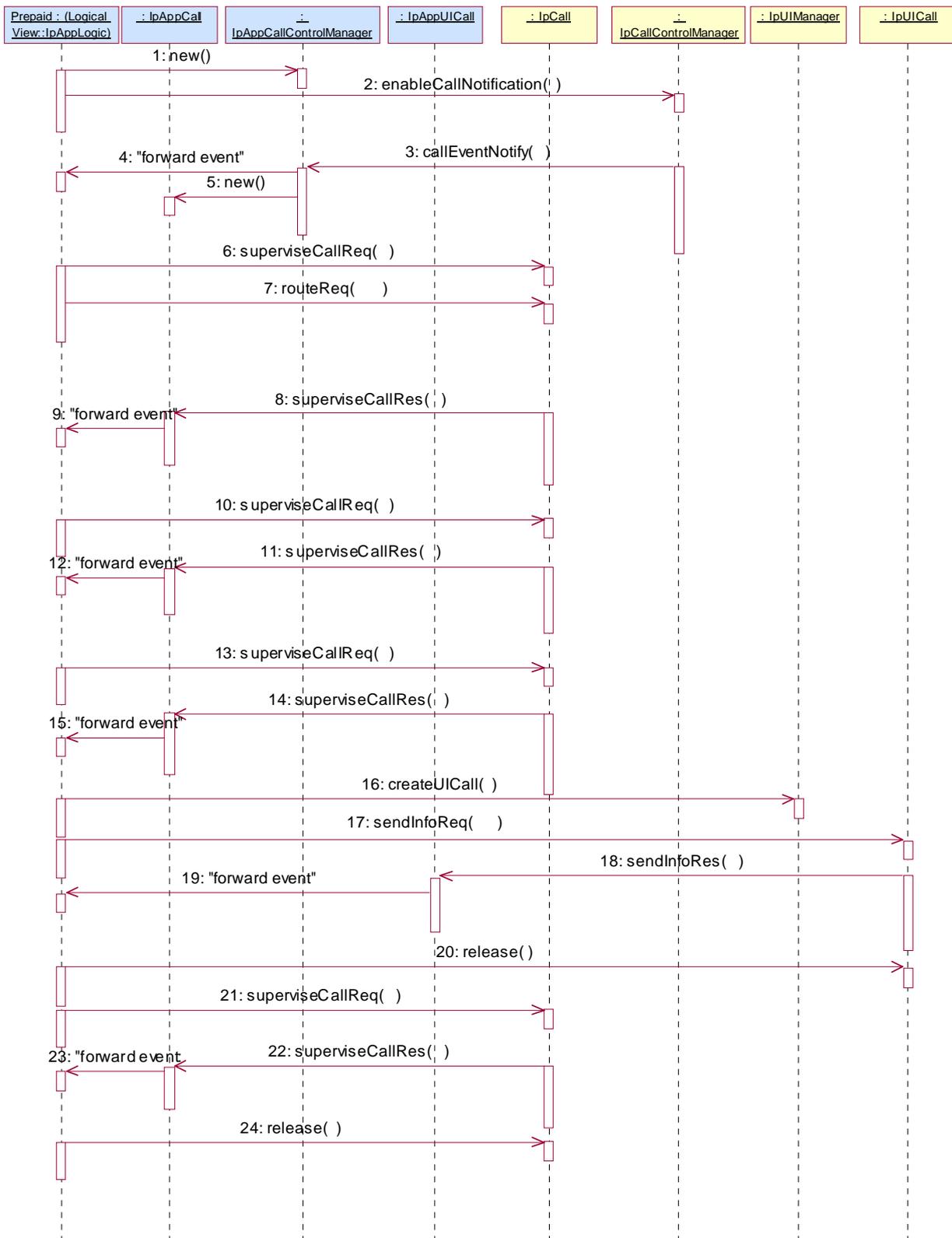
The following sequence diagram shows how an application can receive notifications that have not been created by the application, but are provisioned from within the network.



- 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 IpMultiPartyCallControlManager interface. From now on the gateway will not send any notifications to the application that are created in the network.

## 5.4 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.



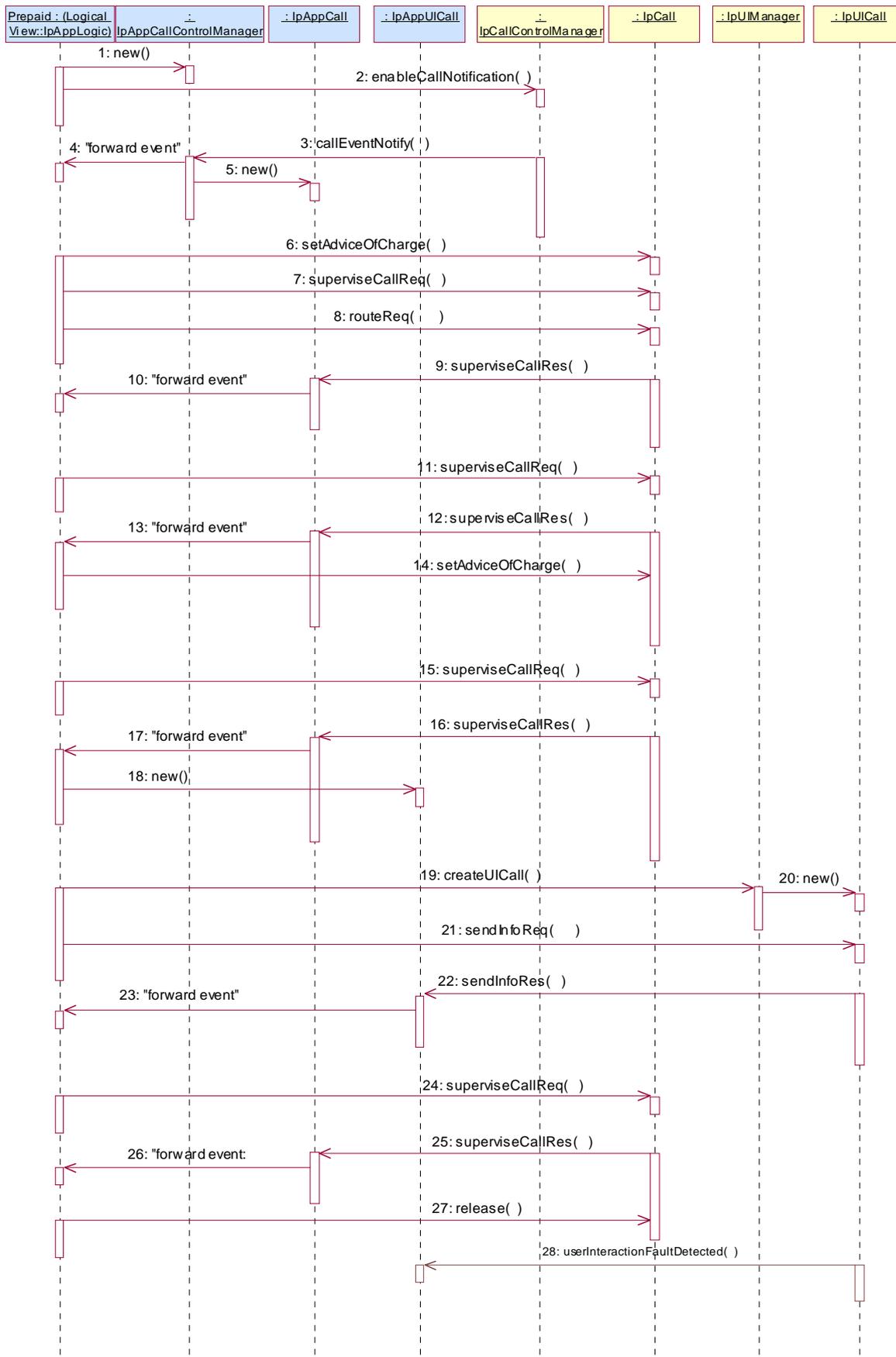
- 1: This message is used by the application to create an object implementing the IpAppCallControlManager interface.
- 2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within 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.

## 5.5 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.



- 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.

---

## 6 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.

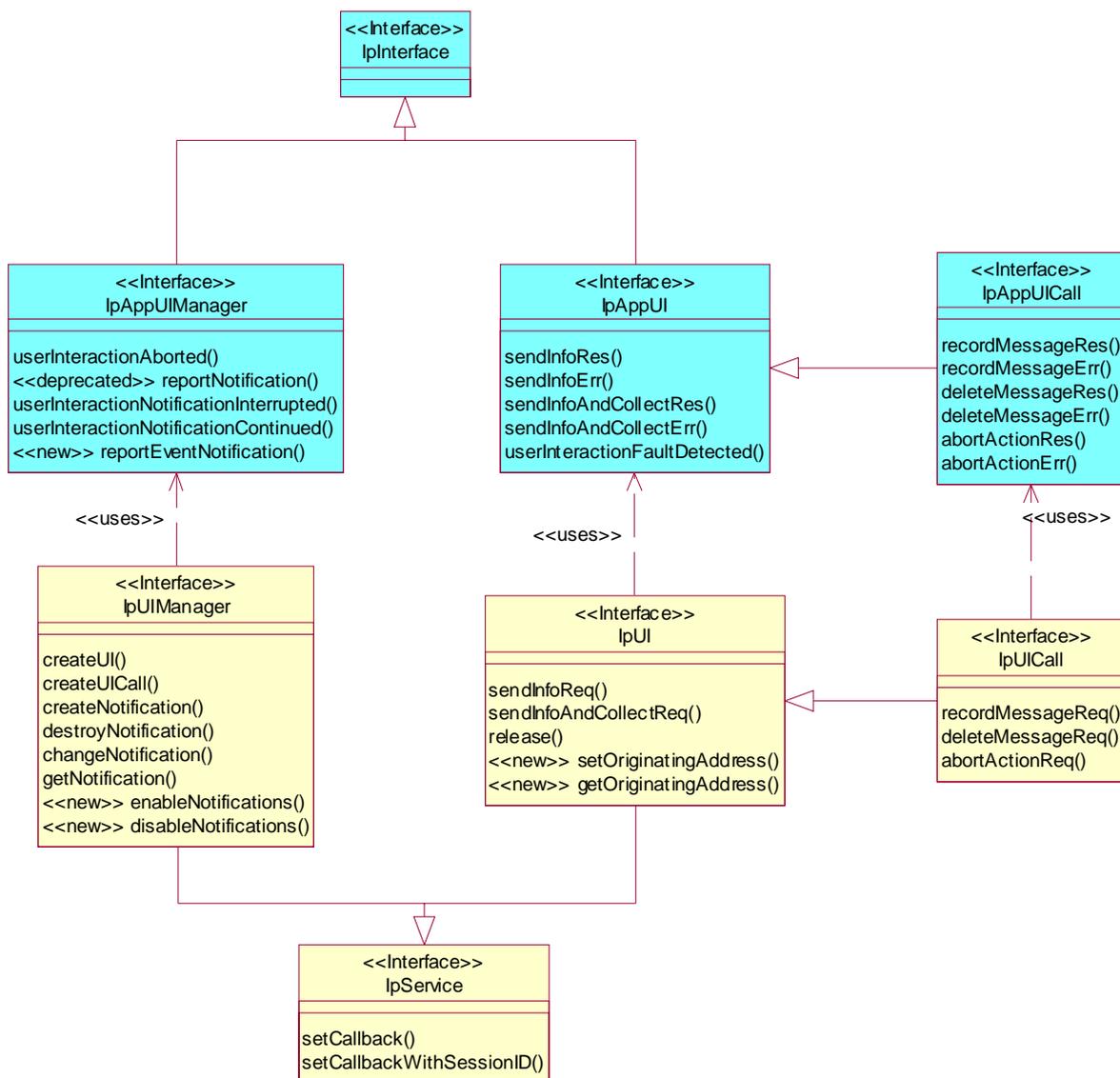


Figure: Generic User Interaction Package Overview

## 7 The Service Interface Specifications

### 7.1 Interface Specification Format

This section 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.

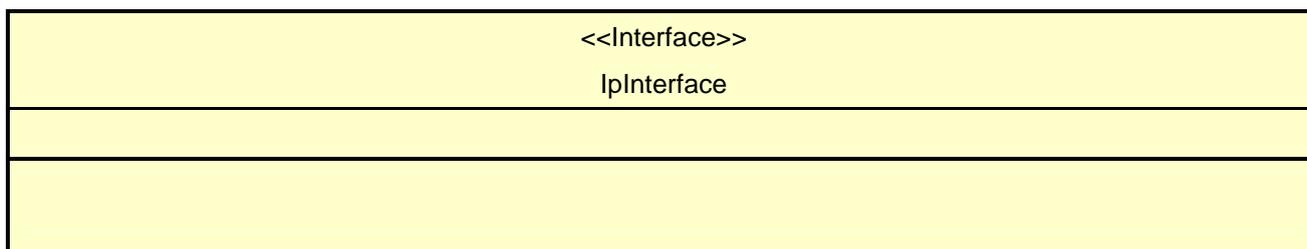
## 7.1.4 State Model

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.

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

#### 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.

##### *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.

##### *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

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

## 8.1 Interface Class IpUIManager

Inherits from: IpService.

This interface is the 'service manager' interface for the Generic User Interaction Service and provides the management functions to the Generic User Interaction Service.

This interface shall be implemented by a Generic User Interaction SCF. The createUI() method, or the createUICall() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications() and disableNotifications() methods shall be implemented as a minimum requirement.

<<Interface>> IpUIManager
<pre> createUI (appUI : in IpAppUIRef, userAddress : in TpAddress) : TpUIIdentifier createUICall (appUI : in IpAppUICallRef, uiTargetObject : in TpUITargetObject) : TpUICallIdentifier createNotification (appUIManager : in IpAppUIManagerRef, eventCriteria : in TpUIEventCriteria) :   TpAssignmentID destroyNotification (assignmentID : in TpAssignmentID) : void changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpUIEventCriteria) : void getNotification () : TpUIEventCriteriaResultSet &lt;&lt;new&gt;&gt; enableNotifications (appUIManager : in IpAppUIManagerRef) : TpAssignmentID &lt;&lt;new&gt;&gt; disableNotifications () : void           </pre>

### 8.1.1 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**

### 8.1.2 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**

### 8.1.3 Method createNotification()

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

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

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

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**

### 8.1.4 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**

### 8.1.5 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**

### 8.1.6 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**

### 8.1.7 Method <<new>> enableNotifications()

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

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

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

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

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

#### *Parameters*

**appUIManager : in IpAppUIManagerRef**

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

#### *Returns*

**TpAssignmentID**

#### *Raises*

**TpCommonExceptions**

### 8.1.8 Method <<new>> disableNotifications()

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

#### *Parameters*

No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

## 8.2 Interface Class IpAppUIManager

Inherits from: IpInterface.

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

<<Interface>> IpAppUIManager
userInteractionAborted (userInteraction : in TpUIIdentifier) : void <<deprecated>> reportNotification (userInteraction : in TpUIIdentifier, eventInfo : in TpUIEventInfo, assignmentID : in TpAssignmentID) : IpAppUIRef userInteractionNotificationInterrupted () : void userInteractionNotificationContinued () : void <<new>> reportEventNotification (userInteraction : in TpUIIdentifier, eventNotificationInfo : in TpUIEventNotificationInfo, assignmentID : in TpAssignmentID) : IpAppUIRef

### 8.2.1 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.

### 8.2.2 Method <<deprecated>> reportNotification()

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

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

Returns: appUI

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

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

*Parameters***userInteraction : in TpUIIdentifier**

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

**eventInfo : in TpUIEventInfo**

Specifies data associated with this event.

**assignmentID : in TpAssignmentID**

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

*Returns***IpAppUIRef**

### 8.2.3 Method userInteractionNotificationInterrupted()

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

*Parameters*

No Parameters were identified for this method

### 8.2.4 Method userInteractionNotificationContinued()

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

*Parameters*

No Parameters were identified for this method

### 8.2.5 Method <<new>> reportEventNotification()

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

Returns: appUI

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

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

*Parameters***userInteraction : in TpUIIdentifier**

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

**eventNotificationInfo : in TpUIEventNotificationInfo**

Specifies data associated with this event.

**assignmentID : in TpAssignmentID**

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

*Returns*

**IpAppUIRef**

## 8.3 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.

<<Interface>> IpUI
<pre> sendInfoReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, language : in TpLanguage,   variableInfo : in TpUIVariableInfoSet, repeatIndicator : in TpInt32, responseRequested : in   TpUIResponseRequest) : TpAssignmentID  sendInfoAndCollectReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, language : in   TpLanguage, variableInfo : in TpUIVariableInfoSet, criteria : in TpUICollectCriteria, responseRequested :   in TpUIResponseRequest) : TpAssignmentID  release (userInteractionSessionID : in TpSessionID) : void  &lt;&lt;new&gt;&gt; setOriginatingAddress (userInteractionSessionID : in TpSessionID, origin : in TpString) : void  &lt;&lt;new&gt;&gt; getOriginatingAddress (userInteractionSessionID : in TpSessionID) : TpString </pre>

### 8.3.1 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.

**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**

### 8.3.2 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.

**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.

**responseRequested : in TpUIResponseRequest**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_NETWORK\_STATE, P\_ILLEGAL\_ID, P\_ID\_NOT\_FOUND, P\_ILLEGAL\_RANGE, P\_INVALID\_COLLECTION\_CRITERIA**

### 8.3.3 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**

### 8.3.4 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**

### 8.3.5 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**

## 8.4 Interface Class IpAppUI

Inherits from: IpInterface.

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

<<Interface>> IpAppUI
sendInfoRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport) : void
sendInfoErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void
sendInfoAndCollectRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport, collectedInfo : in TpString) : void
sendInfoAndCollectErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void
userInteractionFaultDetected (userInteractionSessionID : in TpSessionID, fault : in TpUIFault) : void

### 8.4.1 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\_UICALL\_RESPONSE\_REQUIRED.

#### *Parameters*

**userInteractionSessionID : in TpSessionID**

Specifies the user interaction session ID of the user interaction.

**assignmentID : in TpAssignmentID**

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

**response : in TpUIReport**

Specifies the type of response received from the user.

### 8.4.2 Method sendInfoErr()

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

#### *Parameters*

**userInteractionSessionID : in TpSessionID**

Specifies the user interaction session ID of the user interaction.

**assignmentID : in TpAssignmentID**

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

**error : in TpUIError**

Specifies the error which led to the original request failing.

### 8.4.3 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.

### 8.4.4 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.

### 8.4.5 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.

## 8.5 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.

<<Interface>> IpUICall
<pre> recordMessageReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, criteria : in   TpUIMessageCriteria) : TpAssignmentID deleteMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID abortActionReq (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void         </pre>

### 8.5.1 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**

### 8.5.2 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**

### 8.5.3 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**

## 8.6 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

### 8.6.1 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.

### 8.6.2 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.

### 8.6.3 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.

### 8.6.4 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.

### 8.6.5 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.

### 8.6.6 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.

## 9 State Transition Diagrams

### 9.1 State Transition Diagrams for IpUIManager

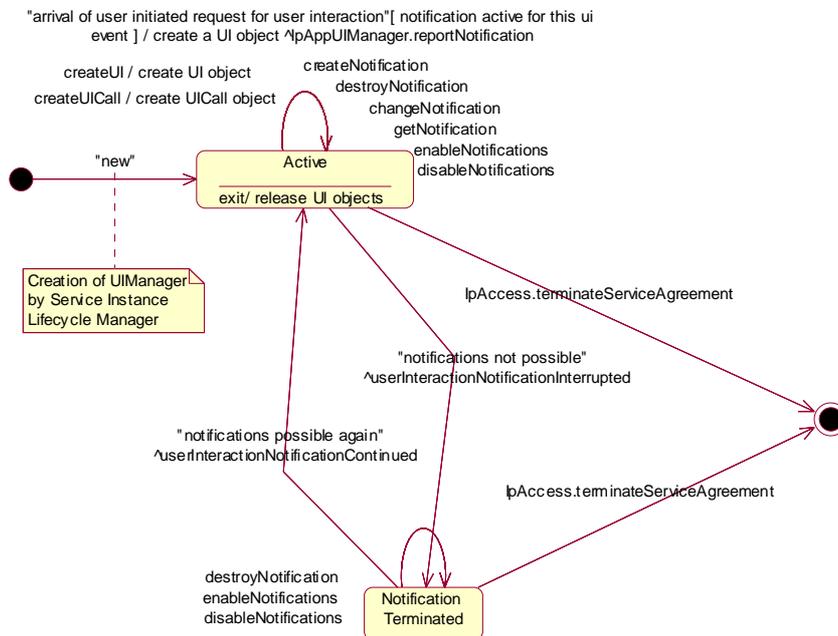


Figure : Application view on the UI Manager

### 9.1.1 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.

### 9.1.2 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.

## 9.2 State Transition Diagrams for IpUI

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

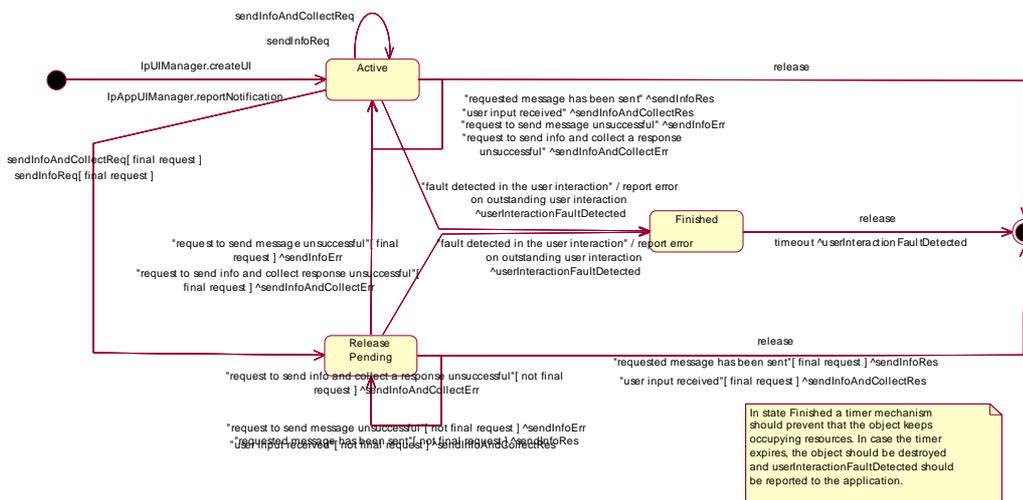


Figure : Application view on the UI object

### 9.2.1 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.

### 9.2.2 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.

### 9.2.3 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.

## 9.3 State Transition Diagrams for IpUICall

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

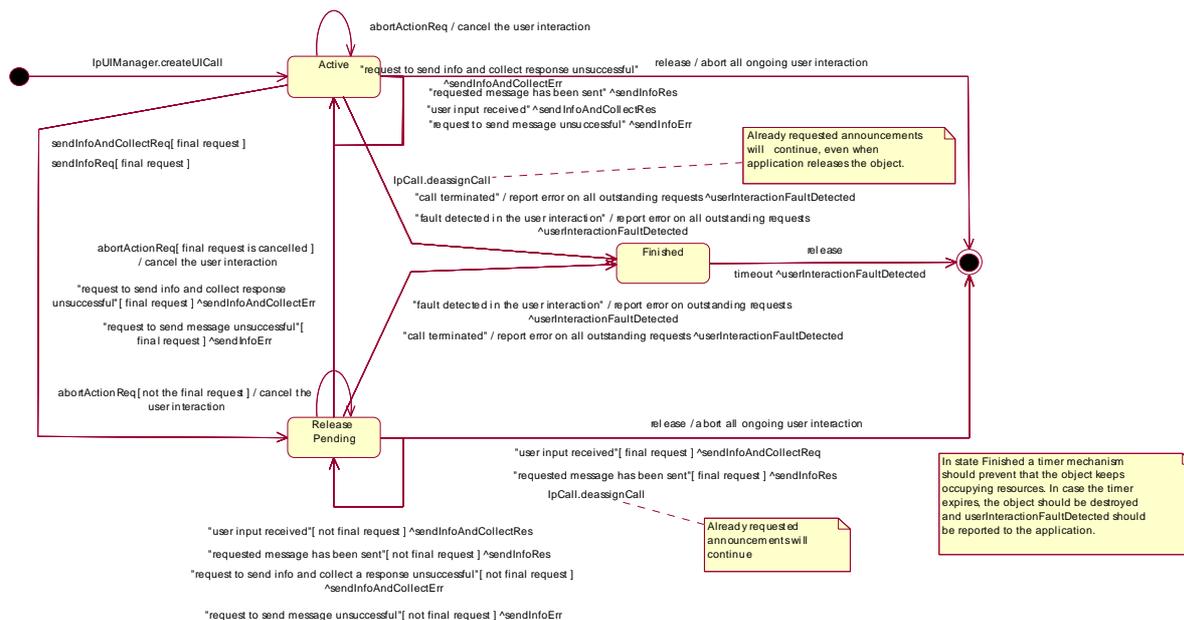


Figure : Application view on the UICall object

### 9.3.1 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.

### 9.3.2 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.

### 9.3.3 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.

---

## 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 value set: {P_INFO_ID, P_URL, P_TEXT}

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_TRIGGERING_ADDRESSES	ADDRESS_RANGE_SET	Specifies which numbers the notification may be set
P_SERVICE_CODE	INTEGER_SET	Specifies the service codes that may be used for notification requests.

---

## 11 Data Definitions

The following data types referenced in this clause are defined in 3GPP TS 29.198-4:

`TpCallIdentifier`  
`TpMultiPartyCallIdentifier`  
`TpCallLegIdentifier`

All other data types referenced but not defined in this clause are common data definitions which may be found in 3GPP TS 29.198-2.

## 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.

Structure Element Name	Structure Element Type
MinLength	TpInt32
MaxLength	TpInt32
EndSequence	TpString
StartTimeout	TpDuration
InterCharTimeout	TpDuration

The structure elements specify the following criteria:

**MinLength:** Defines the minimum number of characters (e.g. digits) to collect.

**MaxLength:** Defines the maximum number of characters (e.g. digits) to collect.

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

**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.

**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.

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 <code>abortActionReq()</code> was called.
P_UI_ERROR_NO_SPACE_AVAILABLE	8	There is no more storage capacity to record the message when the <code>recordMessageReq()</code> 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 TpUIEventInfo

Defines the Sequence of Data Elements that specify a UI 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 DataString.
DataString	TpString	Freely defined data string with a limited length e.g. 160 bytes according to the network policy.

## 11.15 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

## 11.16 TpUIIdentifier

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

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

## 11.17 TpUIInfo

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

Tag Element Type
<a href="#">TpUIInfoType</a>

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

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 or stream to be sent to an end-user's terminal.
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.

## 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 send to an end-user consists of an ID
P_UI_INFO_DATA	1	The information to be send to an end-user consists of a data string
P_UI_INFO_ADDRESS	2	The information to be send 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 a 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

## 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 derived 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

---

## 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 is 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 2919805IDL.ZIP) which accompanies the present document.

---

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

The W3C WSDL representation of this specification is contained in text files (ui interfaces, wsdl and ui\_data.wsdl contained in archive 2919805WSDL.ZIP) which accompanies the present document.

---

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

The Java API realisation of this specification is produced in accordance with the Java Realisation rules defined in Part 1 of this specification series. 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 specification is provided as Java Code, contained in archive 2919805J2SE.ZIP that accompanies the present document.

The J2EE representation of this specification is provided as Java Code, contained in archive 2919805J2EE.ZIP that accompanies the present document.

## Annex D (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010468	002	--	Changing references to JAIN	4.1.0	4.2.0
Dec 2001	CN_14	NP-010598	003	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010598	004	--	Correction of description of sendInfoRes()	4.2.0	4.3.0
Dec 2001	CN_14	NP-010598	005	--	Correction to handling of deassign on related object	4.2.0	4.3.0
Dec 2001	CN_14	NP-010598	006	--	Correction to Exceptions Raised in UI	4.2.0	4.3.0
Dec 2001	CN_14	NP-010598	007	--	Correction to values of TpUIInfoType	4.2.0	4.3.0
Mar 2002	CN_15	NP-020107	008	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	009	--	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020189	010	--	Improve the vague description of P_ID_NOT_FOUND	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	011	--	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020189	012	--	Detach call leg before playing announcement or collecting digits	4.4.0	5.0.0
Jun 2002	CN_16	NP-020189	013	--	Delete P_INVALID_CRITERIA from sendInfoAndCollectReq()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	014	--	Addition of Support for Network Controlled Notifications UI	4.4.0	5.0.0
Jun 2002	CN_16	NP-020189	015	--	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0
Sep 2002	CN_17	NP-020432	018	--	Add text to clarify requirements on support of methods	5.0.0	5.1.0
Sep 2002	CN_17	NP-020432	019	--	Correction on use of NULL in User Interaction API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020432	020	--	Correction to TpUIInfo data type to support binary data for SMS services	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	021		Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030021	023	--	Correction to User Interaction Prepaid Sequence Diagrams	5.1.0	5.2.0
Mar 2003	CN_19	NP-030021	025	--	Correction to getNotification to remove P_INVALID_CRITERIA exception	5.1.0	5.2.0
Mar 2003	CN_19	NP-030021	028	--	Addition of status of methods to User Interaction interfaces	5.1.0	5.2.0
Mar 2003	CN_19	NP-030021	031	--	Corrections to User Interaction	5.1.0	5.2.0
Mar 2003	CN_19	NP-030021	033	--	Correction of User Interaction Event Notification to support non text encodings	5.1.0	5.2.0
Mar 2003	CN_19	NP-030033	029	--	Inconsistent description of use of secondary callback	5.1.0	5.2.0
Jun 2003	CN_20	NP-030238	035	--	Correction of the description for callEventNotify & reportNotification	5.2.0	5.3.0
Jun 2003	CN_20	NP-030244	036	--	Clarify IpUI sendInfoReq()	5.2.0	5.3.0
Jun 2003	CN_20	NP-030244	037	--	Update TpUIInfo for consistency with GMS capabilities	5.2.0	5.3.0
Jun 2003	CN_20	NP-030299	038	1	Specifying the origin of a GUI message	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	039	--	Correction to Java Realisation Annex	5.3.0	5.4.0

---

## History

<b>Document history</b>		
V5.0.0	June 2002	Publication
V5.1.0	September 2002	Publication
V5.2.0	March 2003	Publication
V5.3.0	June 2003	Publication
V5.4.0	September 2003	Publication