SLOVENSKI STANDARD
SIST ES 202 915-4-2 V1.2.1:2005

01-januar-2005

Odprti dostop do storitve (OSA) ï Vmesnik za aplikacijsko programiranje (API) ï 4. del: Krmiljenje klica ï 2. poddel: Generiļ no krmiljenje klica SCF

Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 2: Generic Call Control SCF (Parlay 4)

Ta slovenski standard je istoveten z: ES 202 915-4-2 Version 1.2.1

ICS:

33.040.01	Telekomunikacijski sistemi na splošno	Telecommunication systems in general

SIST ES 202 915-4-2 V1.2.1:2005 **en**

ETSI ES 202 915-4-2 V1.2.1 (2003-08)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 2: Generic Call Control SCF (Parlay 4)

Reference
RES/SPAN-120096-4-2

Keywords
API, IDL, OSA, UML

ETSI

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

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

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

Important notice

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

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

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

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

Contents

Intellectual Property Rights

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

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

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Services and Protocols for Advanced Networks (SPAN).

The present document is part 4, sub-part 2 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 202 915) 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";

Sub-part 2: "Generic Call Control SCF";

Sub-part 3: "Multi-Party Call Control SCF";

Sub-part 4: "Multi-Media Call Control SCF";

Sub-part 5: "Conference Call Control SCF";

Part 5: "User Interaction SCF";

Part 6: "Mobility SCF";

Part 7: "Terminal Capabilities SCF";

Part 8: "Data Session Control SCF";

Part 9: "Generic Messaging SCF";

Part 10: "Connectivity Manager SCF";

Part 11: "Account Management SCF";

Part 12: "Charging SCF";

Part 13: "Policy management SCF";

Part 14: "Presence and Availability Management SCF".

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

The present document forms part of the Parlay 4.1 set of specifications.

The present document is equivalent to 3GPP TS 29.198-4-2 V5.2.0 (Release 5).

1 Scope

The present document is part 4, sub-part 2 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

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

The present document specifies the Generic Call Control Service Capability Feature (SCF) aspects of the interface. All aspects of the Generic Call Control SCF are defined here, these being:

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

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

2 References

The references listed in clause 2 of ES 202 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 202 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 4)".

ETSI ES 202 915-2: "Open Service Access (OSA); Application Programming Interface (API); Part 2: Common Data Definitions (Parlay 4)".

ETSI ES 202 915-4-1: "Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 1: Call Control Common Definitions (Parlay 4)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 202 915-1 apply.

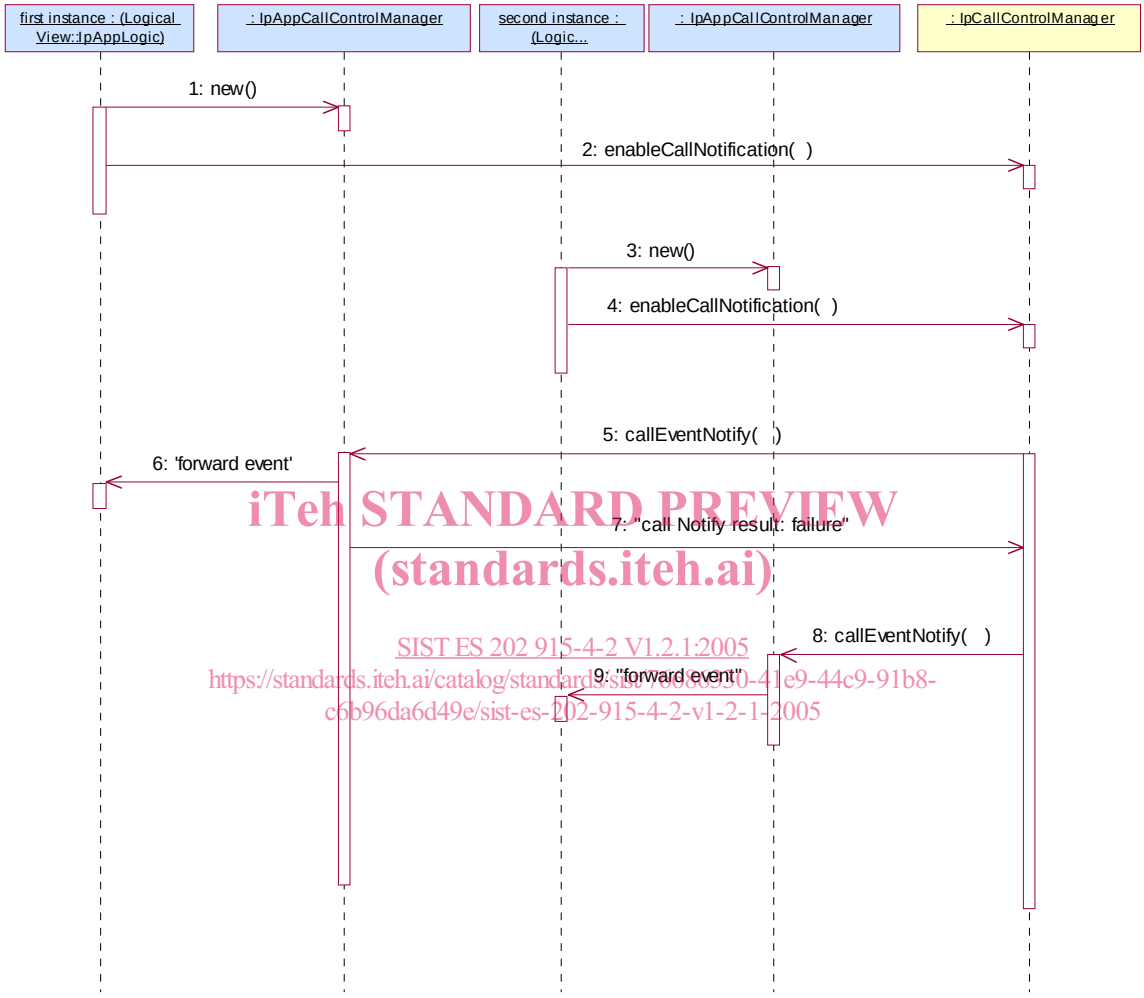
3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 202 915-1 apply.

4 Generic Call Control Service Sequence Diagrams

4.1 Additional Callbacks

The following sequence diagram shows how an application can register two call back interfaces for the same set of events. If one of the call backs can not be used, e.g. because the application crashed, the other call back interface is used instead.



1: The first instance of the application is started on node 1. The application creates a new IpAppCallControlManager to handle callbacks for this first instance of the logic.

2: The enableCallNotification is associated with an applicationID. The call control manager uses the applicationID to decide whether this is the same application.

3: The second instance of the application is started on node 2. The application creates a new IpAppCallControlManager to handle callbacks for this second instance of the logic.

4: The same enableCallNotification request is sent as for the first instance of the logic. Because both requests are associated with the same application, the second request is not rejected, but the specified callback object is stored as an additional callback.

5: When the trigger occurs one of the first instance of the application is notified. The gateway may have different policies on how to handle additional callbacks, e.g., always first try the first registered or use some kind of round robin scheme.

6: The event is forwarded to the first instance of the logic.

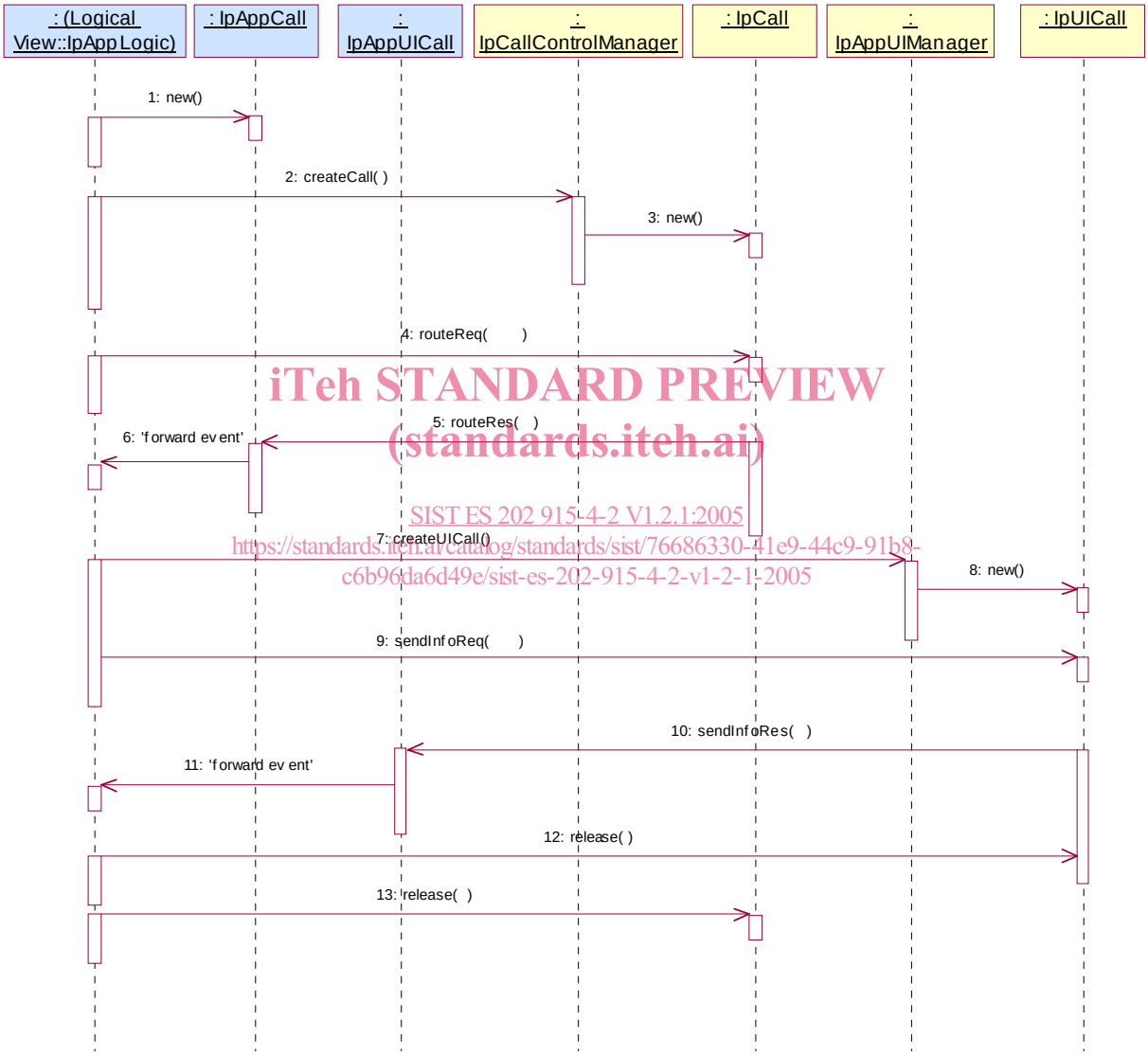
7: When the first instance of the application is overloaded or unavailable this is communicated with an exception to the call control manager.

8: Based on this exception the call control manager will notify another instance of the application (if available).

9: The event is forwarded to the second instance of the logic.

4.2 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) is 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.

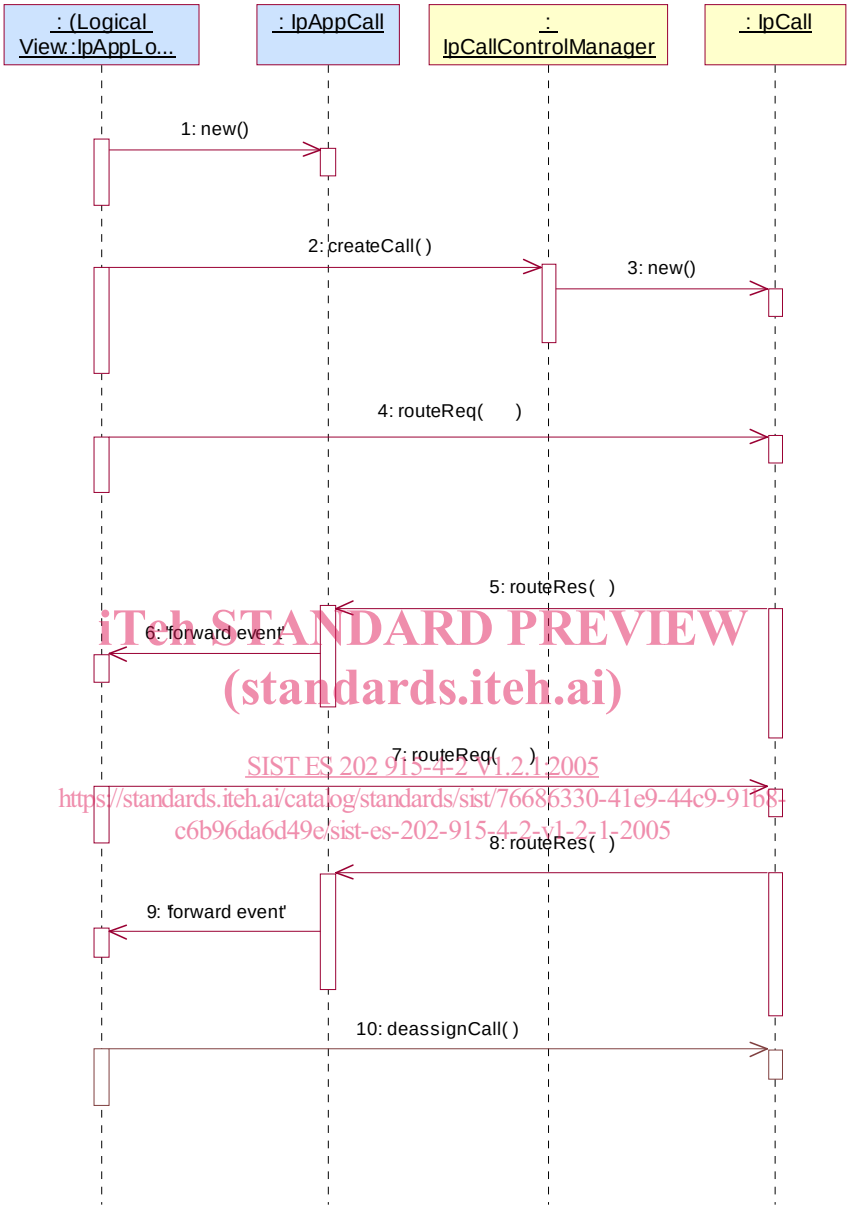
11: The event is forwarded to the application logic.

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.

4.3 Application Initiated Call

The following sequence diagram shows an application creating a call between party A and party B. This sequence could be done after a customer has accessed a Web page and selected a name on the page of a person or organisation to talk to.



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) is met, it is created.

4: This message is used to route the call to the A subscriber (origination). In the message the application request response when the A party answers.

5: This message indicates that the A party answered the call.

6: This message forwards the previous message to the application logic.

7: This message is used to route the call to the B-party. Also in this case a response is requested for call answer or failure.

8: This message indicates that the B-party answered the call. The call now has two parties and a speech connection is automatically established between them.

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

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

4.4 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.