



SLOVENSKI STANDARD

SIST ES 202 915-4-3 V1.2.2:2005

01-januar-2005

CXdfhj`Xcglcd`Xc`glcf]hj Y`fCG5L`E`Ja Ygb]_`nUd`_]_UW`g_c`dfc[fUa JfUb`Y`f5 D`L`E`(" XY.`?fa J`Yb`Y`_]WU`E`" "dcXXY.`?fa J`Yb`Y`_]WU`j Y`i XY`Y`YbW`j`G7:

Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 4)

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Ta slovenski standard je istoveten z: **ES 202 915-4-3 Version 1.2.2**

SIST ES 202 915-4-3 V1.2.2:2005
<https://standards.iteh.ai/catalog/standards/sist/53853781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005>

ICS:

33.040.01	Telekomunikacijski sistemi na splošno	Telecommunication systems in general
-----------	--	---

SIST ES 202 915-4-3 V1.2.2:2005 **en**

iTeh STANDARD PREVIEW **(standards.iteh.ai)**

SIST ES 202 915-4-3 V1.2.2:2005

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005>

ETSI ES 202 915-4-3 V1.2.2 (2003-08)

ETSI Standard

**Open Service Access (OSA);
Application Programming Interface (API);
Part 4: Call Control;
Sub-part 3: Multi-Party Call Control SCF
(Parlay 4)**



iTeh STANDARD PREVIEW
(standards.iteh.ai)

[SIST ES 202 915-4-3 V1.2.2:2005](https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005)

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005>



ReferenceRES/SPAN-120105

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

iTeh STANDARD PREVIEW
(standards.iteh.ai)

SIST ES 202 915-4-3 V1.2.2:2005

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02e1d/etsi-es-202-915-4-3-v1-2-2-2005>

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

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.

© The Parlay Group 2003.

All rights reserved.

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

Contents

Intellectual Property Rights	7
Foreword.....	7
1 Scope	9
2 References	9
3 Definitions and abbreviations.....	9
3.1 Definitions	9
3.2 Abbreviations	9
4 MultiParty Call Control Service Sequence Diagrams	10
4.1 Application initiated call setup.....	10
4.2 Call Barring 2	11
4.3 Call forwarding on Busy Service	13
4.4 Call Information Collect Service.....	14
4.5 Complex Card Service.....	17
4.6 Hotline Service	20
4.7 Network Controlled Notifications	23
4.8 Use of the Redirected event.....	24
5 Class Diagrams	24
6 MultiParty Call Control Service Interface Classes	26
6.1 Interface Class IpMultiPartyCallControlManager	26
6.1.1 Method createCall()	27
6.1.2 Method createNotification()	27
6.1.3 Method destroyNotification()	28
6.1.4 Method changeNotification()	29
6.1.5 Method <<deprecated>> getNotification()	29
6.1.6 Method setCallLoadControl()	29
6.1.7 Method <<new>> enableNotifications()	30
6.1.8 Method <<new>> disableNotifications()	31
6.1.9 Method <<new>> getNextNotification()	31
6.2 Interface Class IpAppMultiPartyCallControlManager	31
6.2.1 Method reportNotification()	32
6.2.2 Method callAborted()	33
6.2.3 Method managerInterrupted()	33
6.2.4 Method managerResumed()	33
6.2.5 Method callOverloadEncountered()	33
6.2.6 Method callOverloadCeased()	33
6.3 Interface Class IpMultiPartyCall	34
6.3.1 Method getCallLegs()	34
6.3.2 Method createCallLeg()	35
6.3.3 Method createAndRouteCallLegReq()	35
6.3.4 Method release()	36
6.3.5 Method deassignCall()	36
6.3.6 Method getInfoReq()	37
6.3.7 Method setChargePlan()	37
6.3.8 Method setAdviceOfCharge()	38
6.3.9 Method superviseReq()	38
6.4 Interface Class IpAppMultiPartyCall	38
6.4.1 Method getInfoRes()	39
6.4.2 Method getInfoErr()	39
6.4.3 Method superviseRes()	40
6.4.4 Method superviseErr()	40
6.4.5 Method callEnded()	40
6.4.6 Method createAndRouteCallLegErr()	41
6.5 Interface Class IpCallLeg	41

6.5.1	Method routeReq()	42
6.5.2	Method eventReportReq()	43
6.5.3	Method release()	43
6.5.4	Method getInfoReq()	44
6.5.5	Method getCall()	44
6.5.6	Method attachMediaReq()	44
6.5.7	Method detachMediaReq()	45
6.5.8	Method getCurrentDestinationAddress()	45
6.5.9	Method continueProcessing()	45
6.5.10	Method setChargePlan()	46
6.5.11	Method setAdviceOfCharge()	46
6.5.12	Method superviseReq()	46
6.5.13	Method deassign()	47
6.6	Interface Class IpAppCallLeg	47
6.6.1	Method eventReportRes()	48
6.6.2	Method eventReportErr()	49
6.6.3	Method attachMediaRes()	49
6.6.4	Method attachMediaErr()	49
6.6.5	Method detachMediaRes()	49
6.6.6	Method detachMediaErr()	50
6.6.7	Method getInfoRes()	50
6.6.8	Method getInfoErr()	50
6.6.9	Method routeErr()	50
6.6.10	Method superviseRes()	51
6.6.11	Method superviseErr()	51
6.6.12	Method callLegEnded()	51
7	MultiParty Call Control Service State Transition Diagrams	52
7.1	State Transition Diagrams for IpMultiPartyCallControlManager	52
7.1.1	Active State	52
7.1.2	Interrupted State	52
7.1.3	Overview of allowed methods	53
7.2	State Transition Diagrams for IpMultiPartyCall	53
7.2.1	IDLE State	54
7.2.2	ACTIVE State	54
7.2.3	RELEASED State	54
7.2.4	Overview of allowed methods	54
7.3	State Transition Diagrams for IpCallLeg	54
7.3.1	Originating Call Leg	55
7.3.1.1	Initiating State	56
7.3.1.2	Analysing State	57
7.3.1.3	Active State	59
7.3.1.4	Releasing State	60
7.3.1.5	Overview of allowed methods, Originating Call Leg STD	62
7.3.2	Terminating Call Leg	63
7.3.2.1	Idle (terminating) State	63
7.3.2.2	Active (terminating) State	64
7.3.2.3	Releasing (terminating) State	67
7.3.2.4	Overview of allowed methods and trigger events, Terminating Call Leg STD	69
8	Multi-Party Call Control Service Properties	70
8.1	List of Service Properties	70
8.2	Service Property values for the CAMEL Service Environment	72
9	Multi-Party Call Control Data Definitions	73
9.1	Event Notification Data Definitions	73
9.2	Multi-Party Call Control Data Definitions	73
9.2.1	IpCallLeg	73
9.2.2	IpCallLegRef	73
9.2.3	IpAppCallLeg	73
9.2.4	IpAppCallLegRef	74
9.2.5	IpMultiPartyCall	74
9.2.6	IpMultiPartyCallRef	74

9.2.7	IpAppMultiPartyCall	74
9.2.8	IpAppMultiPartyCallRef	74
9.2.9	IpMultiPartyCallControlManager	74
9.2.10	IpMultiPartyCallControlManagerRef	74
9.2.11	IpAppMultiPartyCallControlManager	74
9.2.12	IpAppMultiPartyCallControlManagerRef	74
9.2.13	TpAppCallLegRefSet	74
9.2.14	TpMultiPartyCallIdentifier	74
9.2.15	TpAppMultiPartyCallBack	75
9.2.16	TpAppMultiPartyCallBackRefType	75
9.2.17	TpAppCallLegCallBack	75
9.2.18	TpMultiPartyCallIdentifierSet	75
9.2.19	TpCallAppInfo	76
9.2.20	TpCallAppInfoType	76
9.2.21	TpCallAppInfoSet	76
9.2.22	TpCallEventRequest	76
9.2.23	TpCallEventRequestSet	77
9.2.24	TpCallEventType	77
9.2.25	TpAdditionalCallEventCriteria	79
9.2.26	TpCallEventInfo	79
9.2.27	TpCallAdditionalEventInfo	80
9.2.28	TpCallNotificationRequest	80
9.2.29	TpCallNotificationScope	80
9.2.30	TpCallNotificationInfo	80
9.2.31	TpCallNotificationReportScope	81
9.2.32	TpNotificationRequested	81
9.2.33	TpNotificationRequestedSet	81
9.2.34	TpReleaseCause	81
9.2.35	TpReleaseCauseSet	81
9.2.36	TpCallLegIdentifier	81
9.2.37	TpCallLegIdentifierSet	82
9.2.38	TpCallLegAttachMechanism	82
9.2.39	TpCallLegConnectionProperties	82
9.2.40	TpCallLegInfoReport	82
9.2.41	TpCallLegInfoType	83
9.2.42	TpCallLegSuperviseTreatment	83
9.2.43	TpCallHighProbabilityCompletion	83
9.2.44	TpNotificationRequestedSetEntry	83
9.2.45	TpCarrierSet	83
9.2.46	TpCarrier	83
9.2.47	TpCarrierID	84
9.2.48	TpCarrierSelectionField	84
Annex A (normative):	OMG IDL Description of Multi-Party Call Control SCF	85
Annex B (informative):	W3C WSDL Description of Multi-Party Call Control SCF	86
Annex C (informative):	Java API Description of the Call Control SCFs	87
Annex D (informative):	Contents of 3GPP OSA Rel-5 Call Control	88
Annex E (informative):	Record of changes	89
E.1	Interfaces	89
E.1.1	New	89
E.1.2	Deprecated	89
E.1.3	Removed	89
E.2	Methods	90
E.2.1	New	90
E.2.2	Deprecated	90
E.2.3	Modified	90
E.2.4	Removed	90

E.3	Data Definitions	91
E.3.1	New	91
E.3.2	Modified	91
E.3.3	Removed.....	91
E.4	Service Properties.....	91
E.4.1	New	91
E.4.2	Deprecated.....	92
E.4.3	Modified	92
E.4.4	Removed.....	92
E.5	Exceptions	92
E.5.1	New	92
E.5.2	Modified	92
E.5.3	Removed.....	93
E.6	Others	93
History	94

iTeh STANDARD PREVIEW (standards.iteh.ai)

[SIST ES 202 915-4-3 V1.2.2:2005](https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005)

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005>

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 3 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".

ETSI STANDARD PREVIEW
(standards.iteh.ai)

SIST ES 202 915-4-3 V1.2.2:2005

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-14645ad02cdf/sist-es-202-915-4-3-v1-2-2-2005>

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 29.198-4-3 V5.2.0 (Release 5).

iTeh STANDARD PREVIEW (standards.iteh.ai)

SIST ES 202 915-4-3 V1.2.2:2005

<https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005>

1 Scope

The present document is part 4, sub-part 3 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 Multi-Party Call Control Service Capability Feature (SCF) aspects of the interface. All aspects of the Multi-Party 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).

STANDARD PREVIEW
(standards.iteh.ai)

2 References

[SIST ES 202 915-4-3 V1.2.2:2005](https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005)

standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005

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 MultiParty Call Control Service Sequence Diagrams

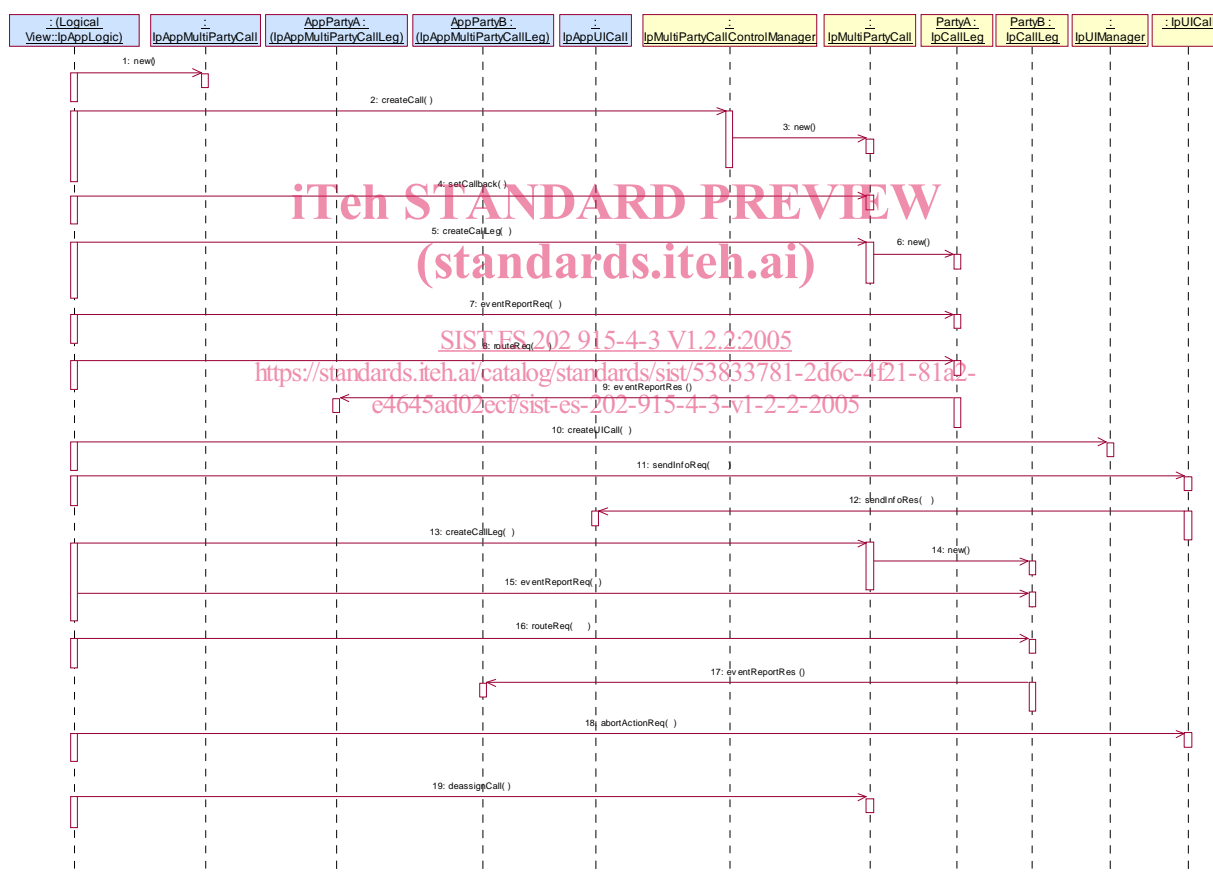
4.1 Application initiated call setup

The following sequence diagram shows an application creating a call between party A and party B. Here, a call is created first. Then party A's call leg is created before events are requested on it for answer and then routed to the call. On answer from Party A, an announcement is played indicating that the call is being set up to party B. While the announcement is being played, party B's call leg is created and then events are requested on it for answer. On answer from Party B the announcement is cancelled and party B is routed to the call.

The service may as a variation be extended to include 3 parties (or more). After the two party call is established, the application can create a new leg and request to route it to a new destination address in order to establish a 3 party call.

The event that causes this to happen could for example be the report of answer event from B-party or controlled by the A-party by entering a service code (mid-call event).

The procedure for call setup to party C is exactly the same as for the set up of the connection to party B (sequence 13 to 17 in the sequence diagram).



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

2: This message requests the object implementing the IpMultiPartyCallControlManager interface to create an object implementing the IpMultiPartyCall interface.

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

4: Once the object implementing the IpMultiPartyCall interface is created it is used to pass the reference of the object implementing the IpAppMultiPartyCall interface as the callback reference to the object implementing the IpMultiPartyCall interface. Note that the reference to the callback interface could already have been passed in the createCall.

- 5: This message instructs the object implementing the IpMultiPartyCall interface to create a call leg for customer A.
- 6: Assuming that the criteria for creating an object implementing the IpCallLeg interface is met, message 6 is used to create it.
- 7: This message requests the call leg for customer A to inform the application when the call leg answers the call.
- 8: The call is then routed to the originating call leg.
- 9: Assuming the call is answered, the object implementing party A's IpCallLeg interface passes the result of the call being answered back to its callback object. This message is then forwarded via another message (not shown) to the object implementing the IpAppLogic interface.
- 10: A UICall object is created and associated with the just created call leg.
- 11: This message is used to inform party A that the call is being routed to party B.
- 12: An indication that the dialogue with party A has commenced is returned via message 13 and eventually forwarded via another message (not shown) to the object implementing the IpAppLogic interface.
- 13: This message instructs the object implementing the IpMultiPartyCall interface to create a call leg for customer B.
- 14: Assuming that the criteria for creating a second object implementing the IpCallLeg interface is met, it is created.
- 15: This message requests the call leg for customer B to inform the application when the call leg answers the call.
- 16: The call is then routed to the call leg.
- 17: Assuming the call is answered, the object implementing party B's IpCallLeg interface passes the result of the call being answered back to its callback object. This message is then forwarded via another message (not shown) to the object implementing the IpAppLogic interface.
- 18: This message then instructs the object implementing the IpUICall interface to stop sending announcements to party A.
- 19: The application deassigns the call. This will also deassign the associated user interaction.

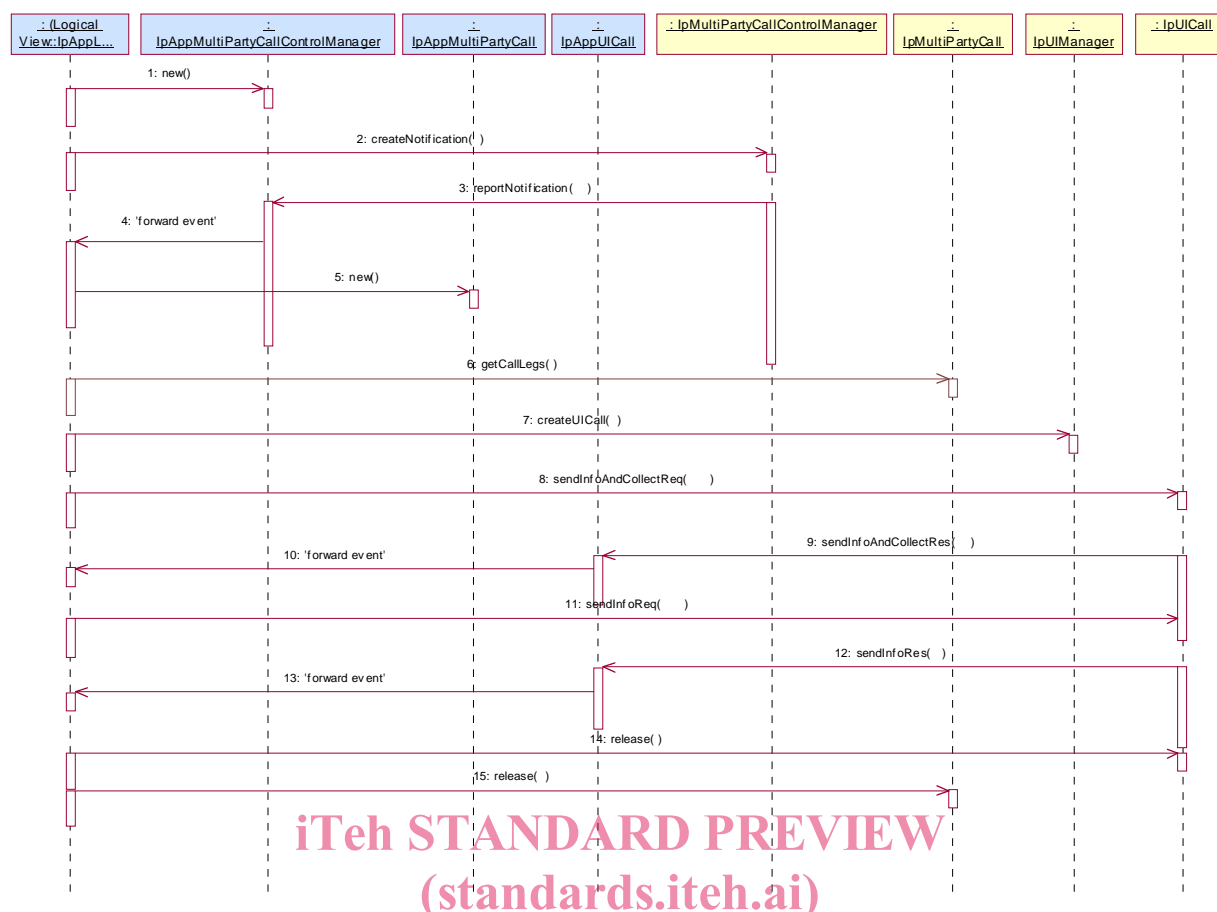
STANDARD PREVIEW
(standards.iteh.ai)

SIST ES 202 915-4-3 V1.2.2:2005

https://standards.iteh.ai/catalog/standards/sist-es-202-915-4-3-v1-2-2-2005
e4645ad02ecf/sist-es-202-915-4-3-v1-2-2-2005

4.2 Call Barring 2

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 rejected and the call is cleared.



iTeh STANDARD PREVIEW
(standards.iteh.ai)

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

[SIST ES 202 915-4-3 V1.2.2:2005](https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-300000000000/sist-202-915-4-3-v1-2-2-2005)

[https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-](https://standards.iteh.ai/catalog/standards/sist/53833781-2d6c-4f21-81a2-300000000000/sist-202-915-4-3-v1-2-2-2005)

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, arrives a message (not shown) is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface (e.g. load control values not exceeded) is 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 IpAppMultiPartyCallControlManager interface.

4: This message is used to forward message 3 to the IpAppLogic.

5: This message is used by the application to create an object implementing the IpAppMultiPartyCall interface. The reference to this object is passed back to the object implementing the IpMultiPartyCallControlManager using the return parameter of the callEventNotify.

6: The application requests a list of all the legs currently in the call.

7: This message is used to create a UICall object that is associated with the incoming leg of the call.

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: Assuming an incorrect PIN is entered, the calling party is informed using additional dialogue of the reason why the call cannot be completed.

12: This message passes the indication that the additional dialogue has been sent.

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

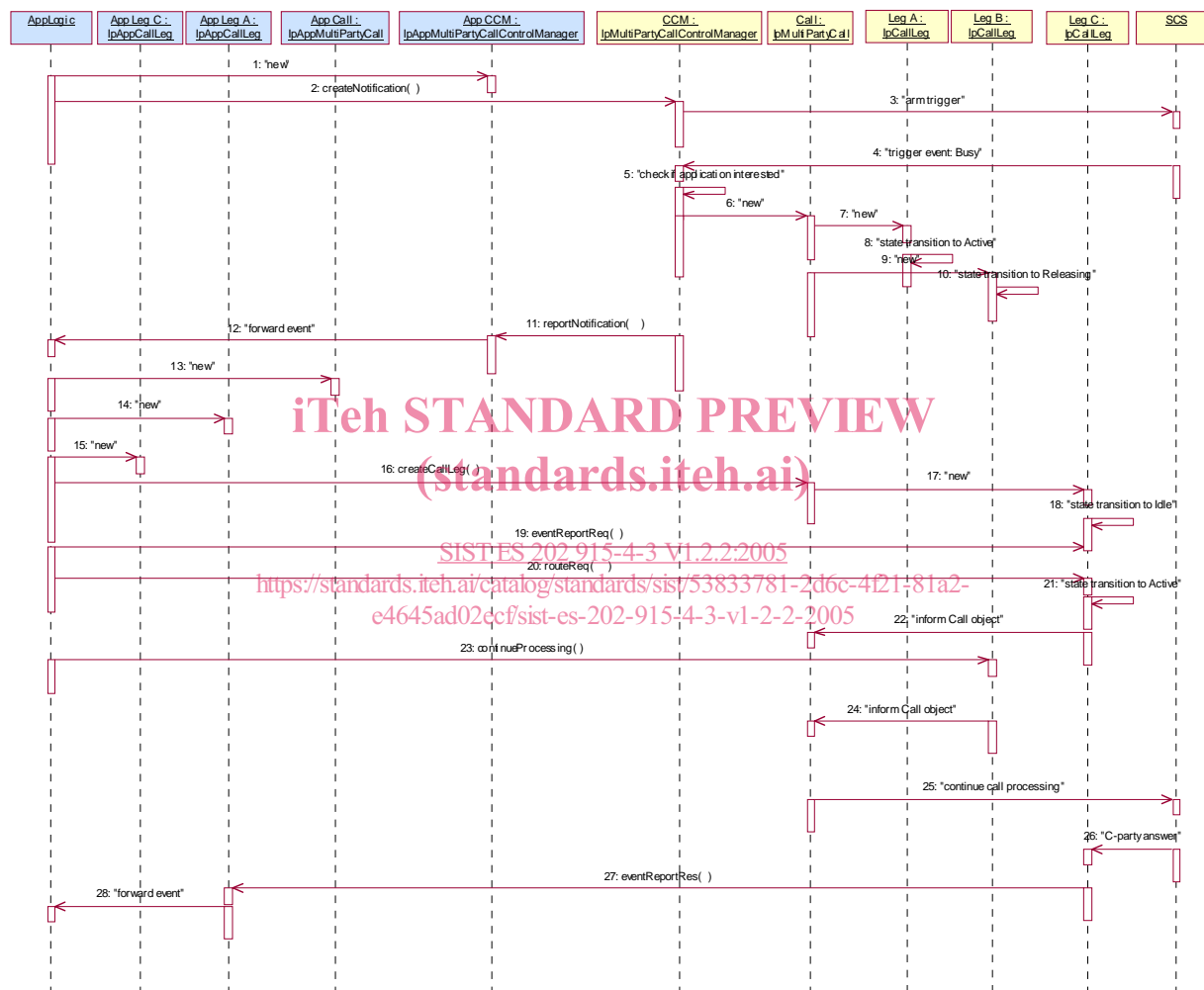
14: No more UI is required, so the UICall object is released.

15: This message is used by the application to clear the call.

4.3 Call forwarding on Busy Service

The following sequence diagram shows an application establishing a call forwarding on busy.

When a call is made from A to B but the B-party is detected to be busy, then the application is informed of this and sets up a connection towards a C party. The C party can for instance be a voicemail system.



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

2: This message is sent by the application to enable notifications on new call events.

4: When a new call, that matches the event criteria, arrives a message ("busy") is directed to the object implementing the IpMultiPartyCallControlManager. Assuming that the criteria for creating an object implementing the IpMultiPartyCall interface is met, other messages are used to create the call and associated call leg objects.

6: A new MultiPartyCall object is created to handle this particular call.

7: A new CallLeg object corresponding to Party A is created.

8: The new Call Leg instance transits to state Active.