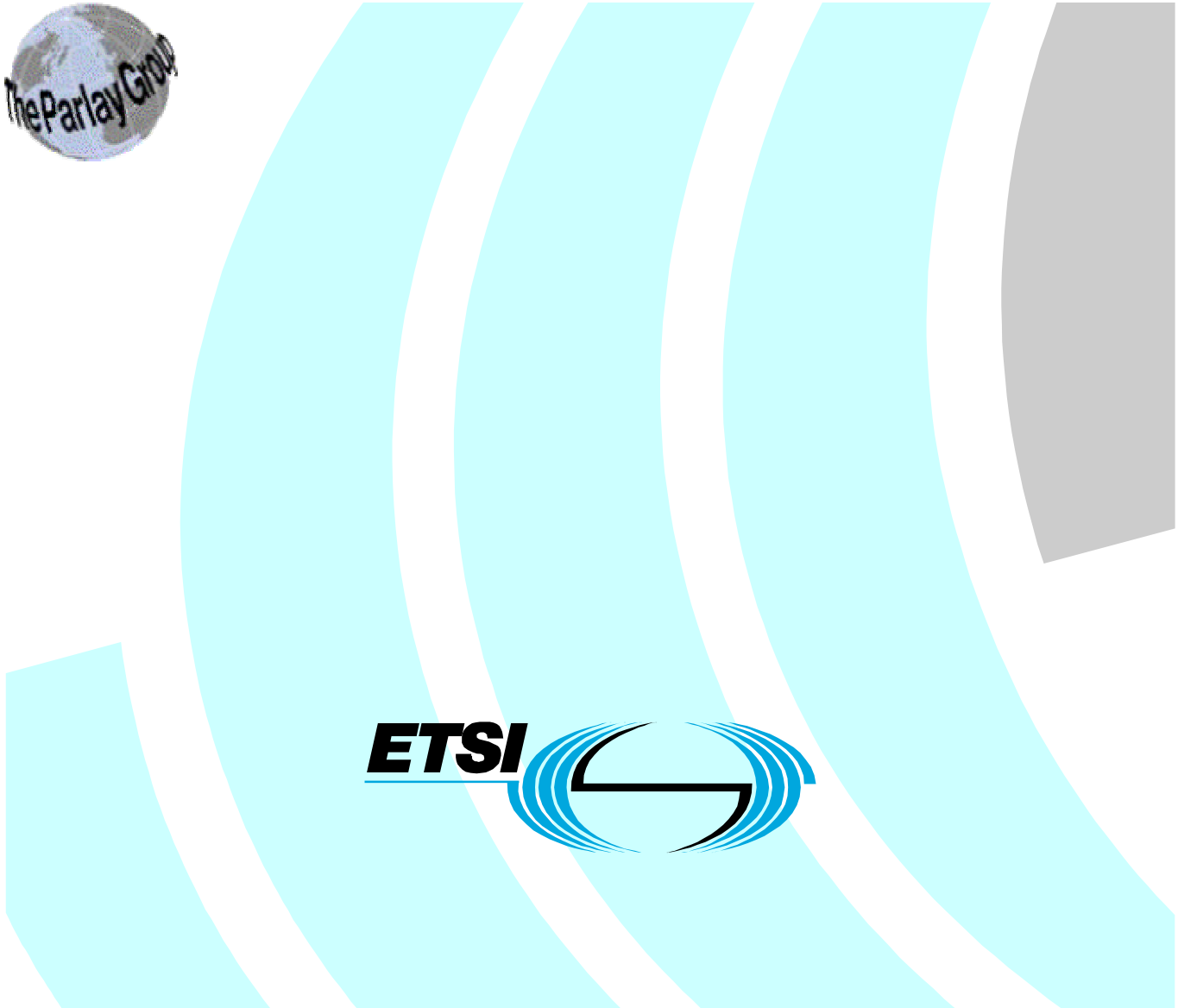
ETSI ES 202 915-4-5 V1.2.1 (2003-08)

ETSI Standard

Open Service Access (OSA);
Application Programming Interface (API);
Part 4: Call Control;
Sub-part 5: Conference Call Control SCF
(Parlay 4)



Reference
RES/SPAN-120096-4-5

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 5 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.

1 Scope

The present document is part 4, sub-part 5 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

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

The present document specifies the Conference Call Control Service Capability Feature (SCF) aspects of the interface. All aspects of the Conference 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.

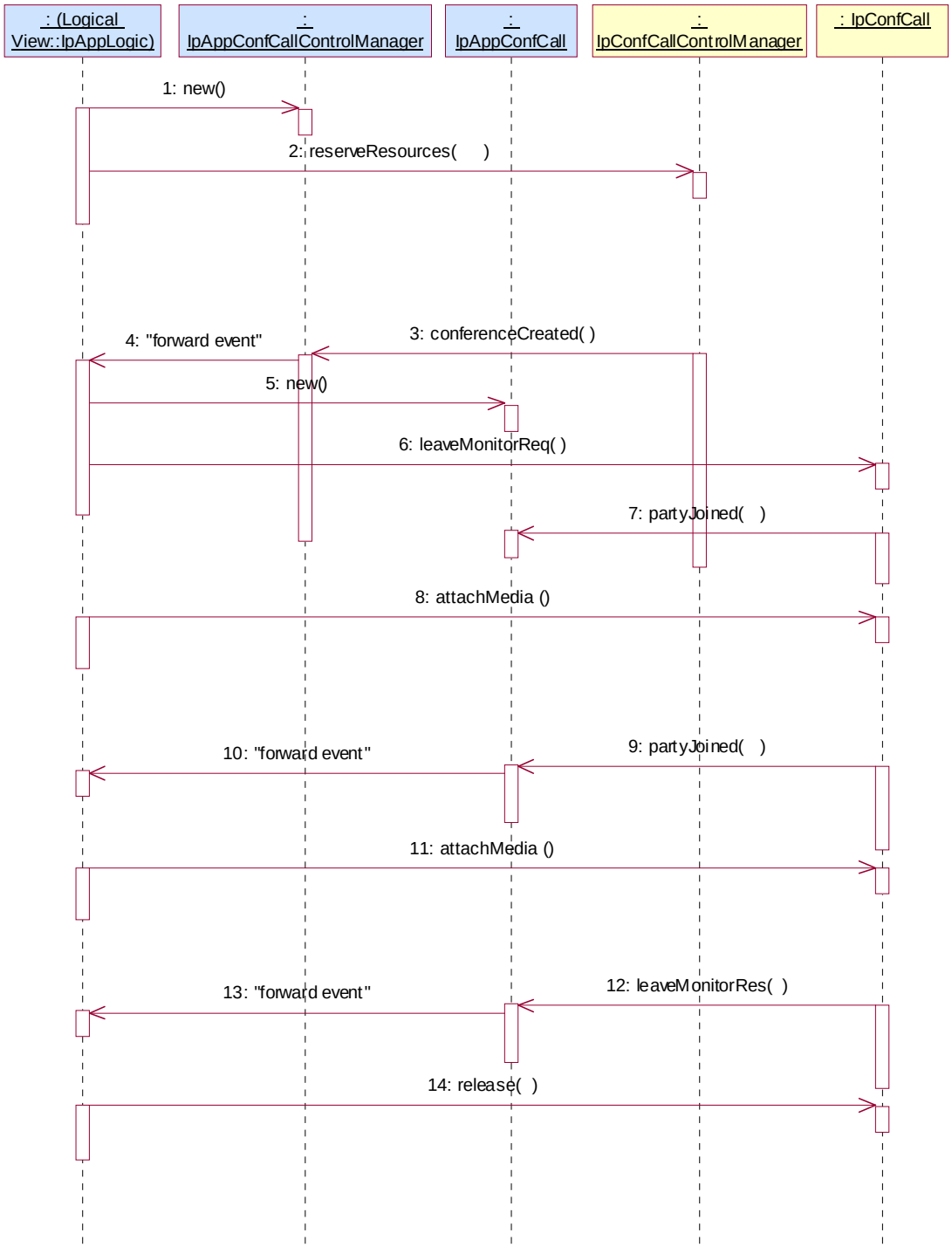
4 Conference Call Control Service Sequence Diagrams

4.1 Meet-me conference without subconferencing

This sequence illustrates a pre-arranged meet-me conference for a specified time period. During this timeslot parties can 'call in to' the meet-me conference by dialling a special number.

For each participant joining the conference, the application can decide to accept the participant in to the conference.

The application can also be notified when parties are leaving the conference.



1: The application creates a new object to receive the callbacks from the conference call control manager.

2: The application reserves resources for some time in the future.

With this same method the application registers interest in the creation of the conference (e.g. when the first party to joins the conference or at the specified start time, this is implementation dependant).

The reservation also includes the conference policy. One of the elements is whether joined parties must be explicitly attached. If so, this is treated as an implicit joinMonitorReq.

3: The conference is created.

4: The message is forwarded to the application.

5: The application creates an object to receive the call back messages from the conference call.

6: The application also requests to be notified when parties leave the conference.

7: The application is notified of the first party that joined the conference

8: When the party is allowed to join the conference, the party is added.

Alternatively, the party could have been rejected with a releaseCallLeg.

9: A new party joins the conference and the application is notified.

10: The message is forwarded to the application.

11: This party also is allowed into the conference by attaching the leg.

12: A party leaves the conference.

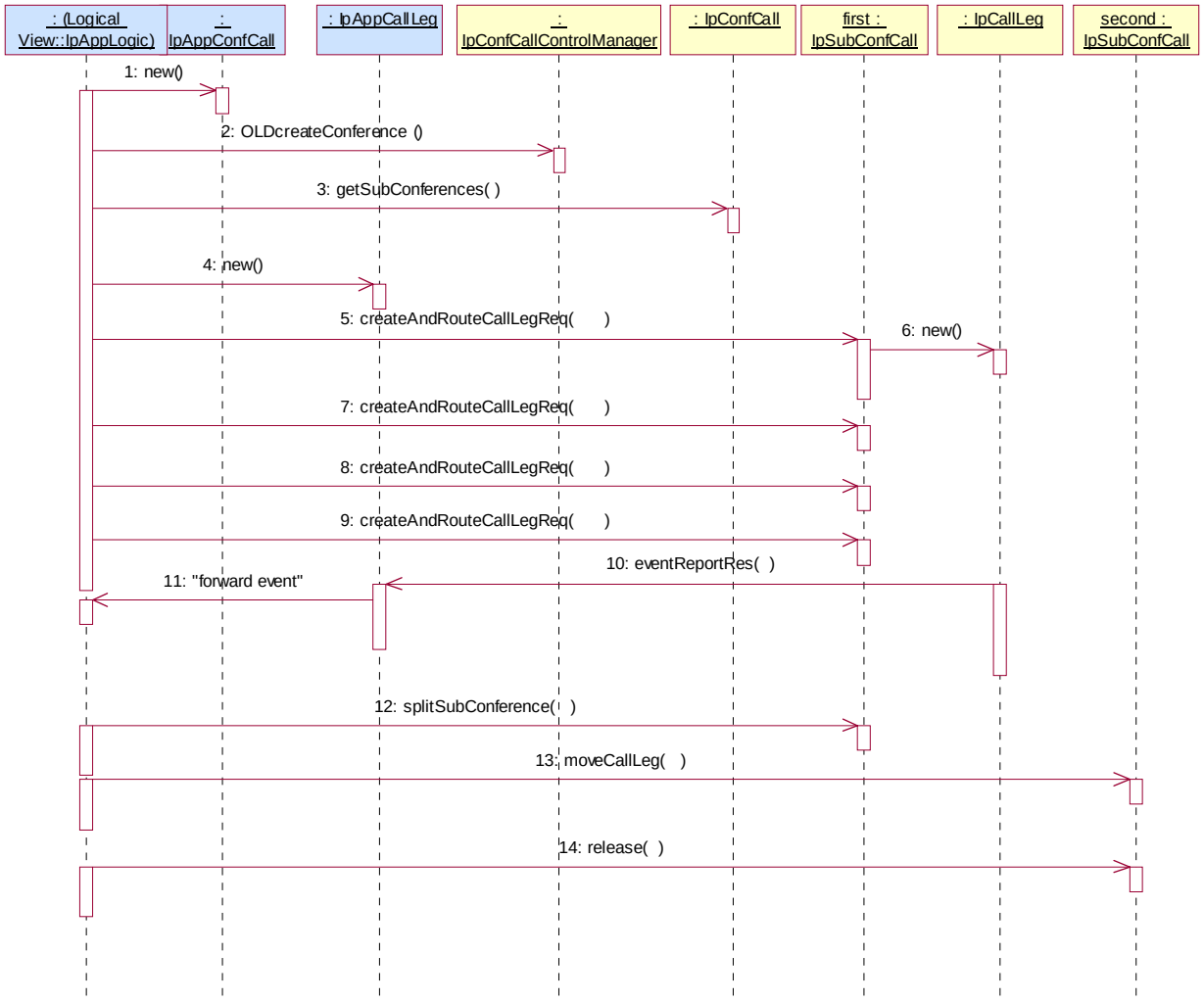
13: The message is forwarded to the application.

14: The application decides to release the entire conference.

4.2 Non-add hoc add-on with subconferencing

This sequence illustrates a prearranged add-on conference. The end user that initiates the call, communicates with the conference application via a web interface (not shown). By dragging and dropping names from the addressbook, the end-users add parties to the conference.

Also via the web-interface, the end-user can group parties in subconferences. Only parties in the same subconference can talk to each other.



1: The application creates a new interface to receive the callbacks from the conference call.

2: The application initiates the conference. There has been no prior resource reservation, so there is a chance that no resources are available when parties are added to the conference.

The conferenceCall interface object is returned.

3: Together with the conference a subconference is implicitly created.

However, the subconference is not returned as a result of the createConference, therefore the application uses this method to get the subconference.

4: The application creates a new IpAppCallLeg interface

5: The application adds the first party to the subconference. This process is repeated for all 4 parties. Note that in the following not all steps are shown.

6: The gateway creates a new IpCallLeg interface.

7: The application adds parties to the subconference.

8: The application adds parties to the subconference.

9: The application adds parties to the subconference.

10: When a party A answers the application is notified.

We assume that all parties answer. This happens in the same way as for party A and is not shown in the following.

11: The message is forwarded to the application.

12: The application decides to split the conference. Party C&D are indicated in the message.

The gateway will create a new subconference and move party C and D to the new subconference.

The configuration is A&B are in speech, C&D are in speech. There is no bearer connection between the two subconferences.

13: The application moves one of the legs from the second subconference back to the first. The configuration now is A,B&C are in speech configuration. D is alone in its own subconference.

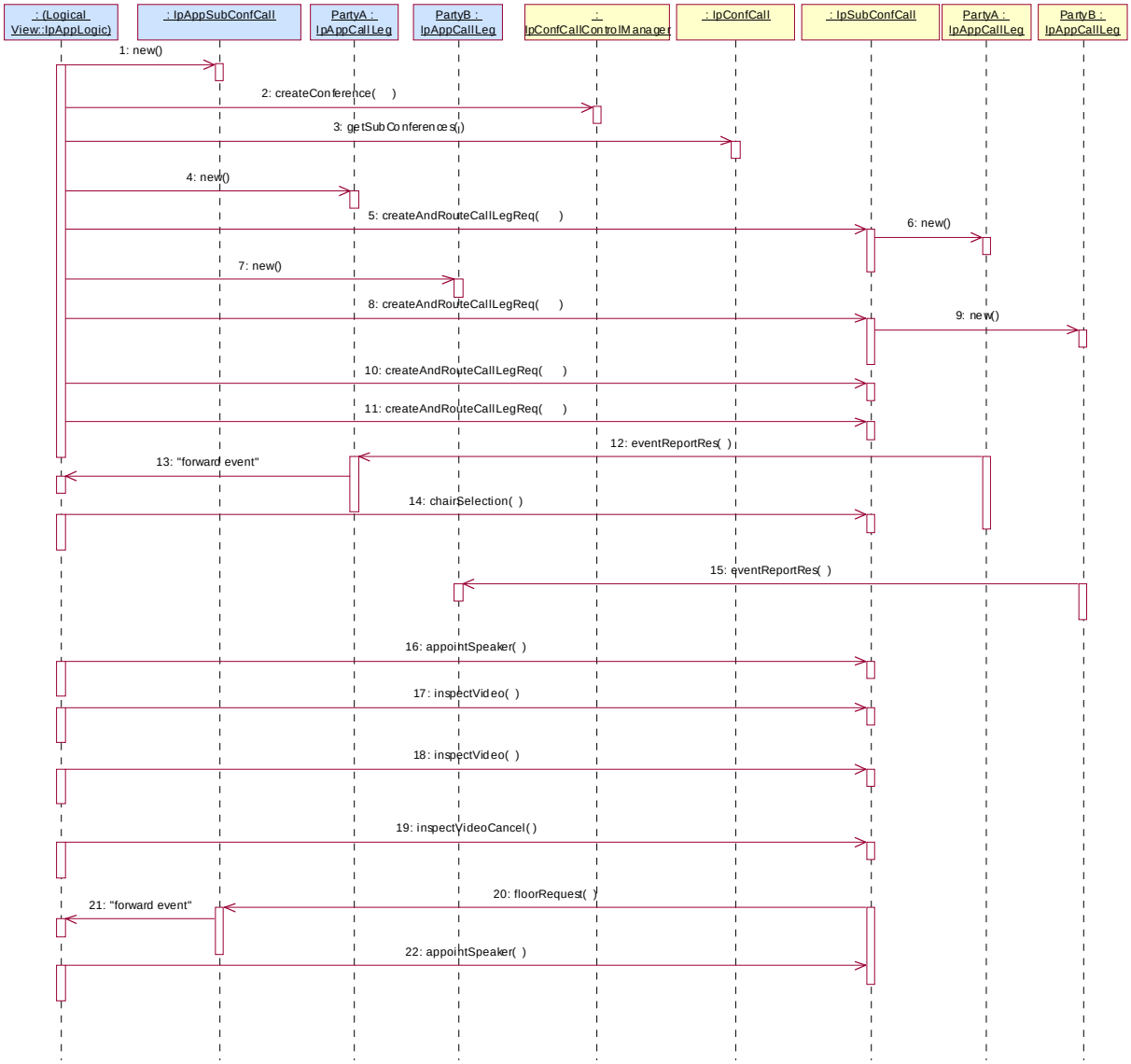
14: The second subconference is released. Since party D was in this subconference, this callleg is also released.

This leaves one subconference with A,B & C.

4.3 Non-addhoc add-on multimedia

This sequence illustrates a prearranged add-on multi-media conference. The end user that initiates the call, communicates with the conference application via a web interface (not shown). By dragging and dropping names from the addressbook, the end-users add parties to the conference.

Also via the web-interface, the end-user can do things that normally the chair would be able to do, e.g., determine who has the floor (e.g., whose video is being broadcast to the other participants) or inspect the video of participants who do not have the floor (e.g., to see how they react to the current speaker).



1: The application creates a new object for receiving callbacks from the MMSubConference.

2: When the user selects the appropriate option in the web interface, the application will create a conference without resource reservation. The policy for video is set to 'chairperson switched.

3: The application requests the subconference that was implicitly created together with the conference.

4: The application creates a new IpAppCallLeg interface.

5: The application adds the first party to the subconference. This process is repeated for all 4 parties. Note that in the following not all steps are shown.

6: The gateway creates a new IpCallLeg interface.

7: The application creates a new IpAppCallLeg interface.

8: The application adds parties to the conference and monitors on success.

9: The gateway creates a new IpCallLeg interface.

10: The application adds parties to the conference and monitors on success.

11: The application adds parties to the conference and monitors on success.

12: When a party A answers the application is notified.

We assume that all parties answer.

14: We assume that A was the initiating party.

The initiating end-user is assigned the chairpersonship.

This message is needed to synchronise the chairpersonship in the application with the MCU chairpersonship, since the chair can also use H.323 messages to control the conference.

15: When a party B answers the application is notified. We assume the other parties answer as well and this is not shown below in the sequence.

16: Chairperson (A) decides via WWW interface that party B is the speaker. This means that the video of B is broadcast to the rest.

17: The chairperson selects the video of C in order to judge their reactions on B's proposal.

18: The chairperson selects the video of D in order to judge their reactions on B's proposal.

19: The chairperson goes back to receiving the broadcasted videostream (B)

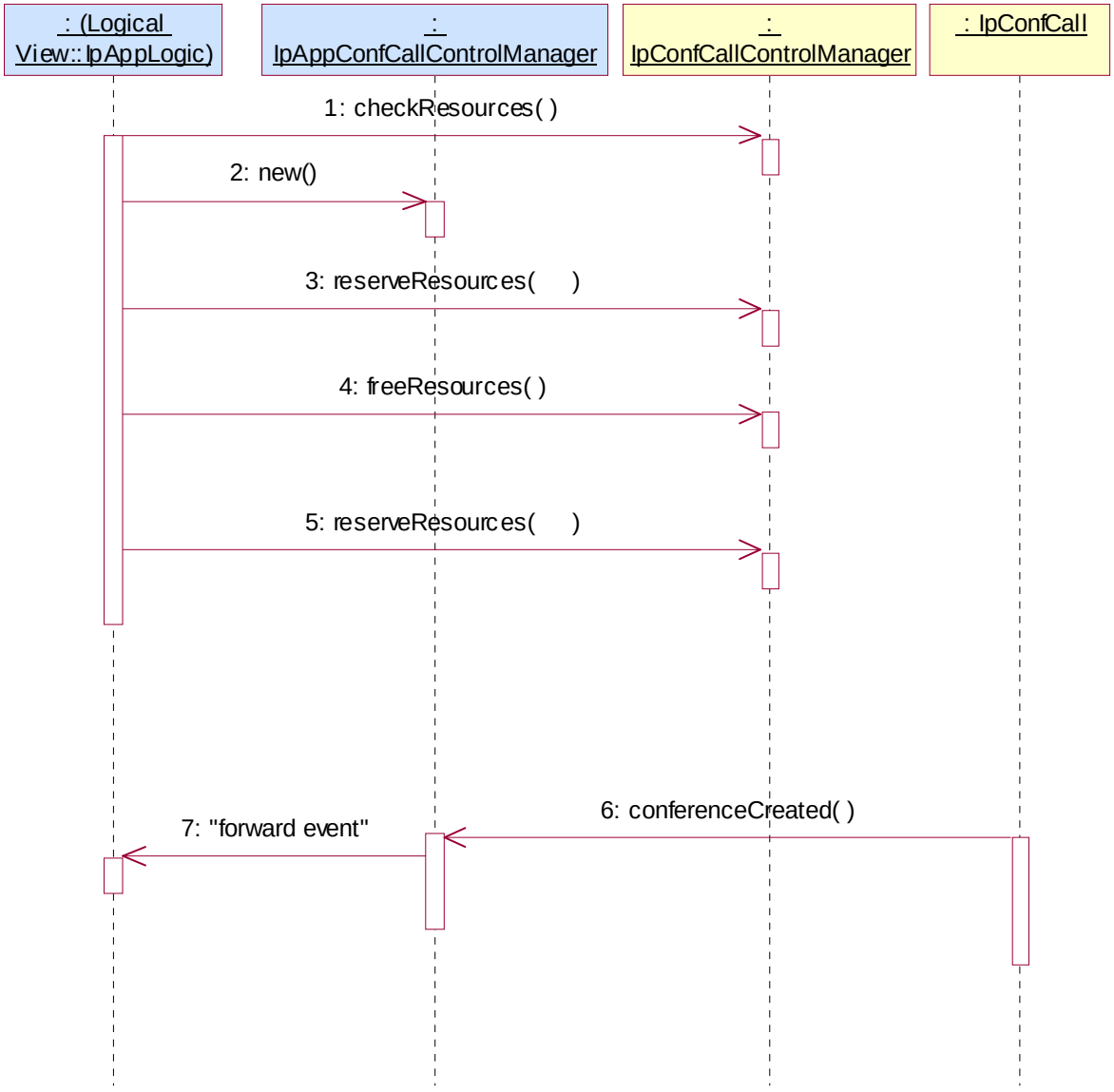
20: User C requests the floor via the H.323 signals. The application is notified of this.

21: The message is forwarded to the application logic.

22: The chairperson (via the WWW interface) grants the request by appointing C as the speaker.

4.4 Resource Reservation

This sequence illustrates how an application can check and reserve resources for a meet-me conference.



1: The application checks if enough conference resources are available in a given time period.

2: The application creates an object to receive callback messages.

3: The application reserves resources for the time period. The callback object is in order to receive a notification when the conference is started.

4: Because the time was wrong by accident, the application cancels the earlier reservation.

5: The application makes a new reservation.

6: At the specified time, or when the first party joins the conference the application is notified.

7: The event is forwarded to the application.

5 Class Diagrams

The conference call control service consists of two packages, one for the interfaces on the application side and one for interfaces on the service side. The class diagrams in the following figures show the interfaces that make up the conference call control application package and the conference call control service package.

This class diagram shows the interfaces that make up the application conference call control service package and the relation to the interfaces in the conference call control service package.

The diagram also shows the inheritance relation between the multi-party call application interfaces and the conference call application interfaces; the conference interfaces are specialisations of the corresponding multi-party call interfaces.

Communication between the application and service packages is done via the <<uses>> relations; the interfaces can communicate with callback methods in the corresponding application interfaces.

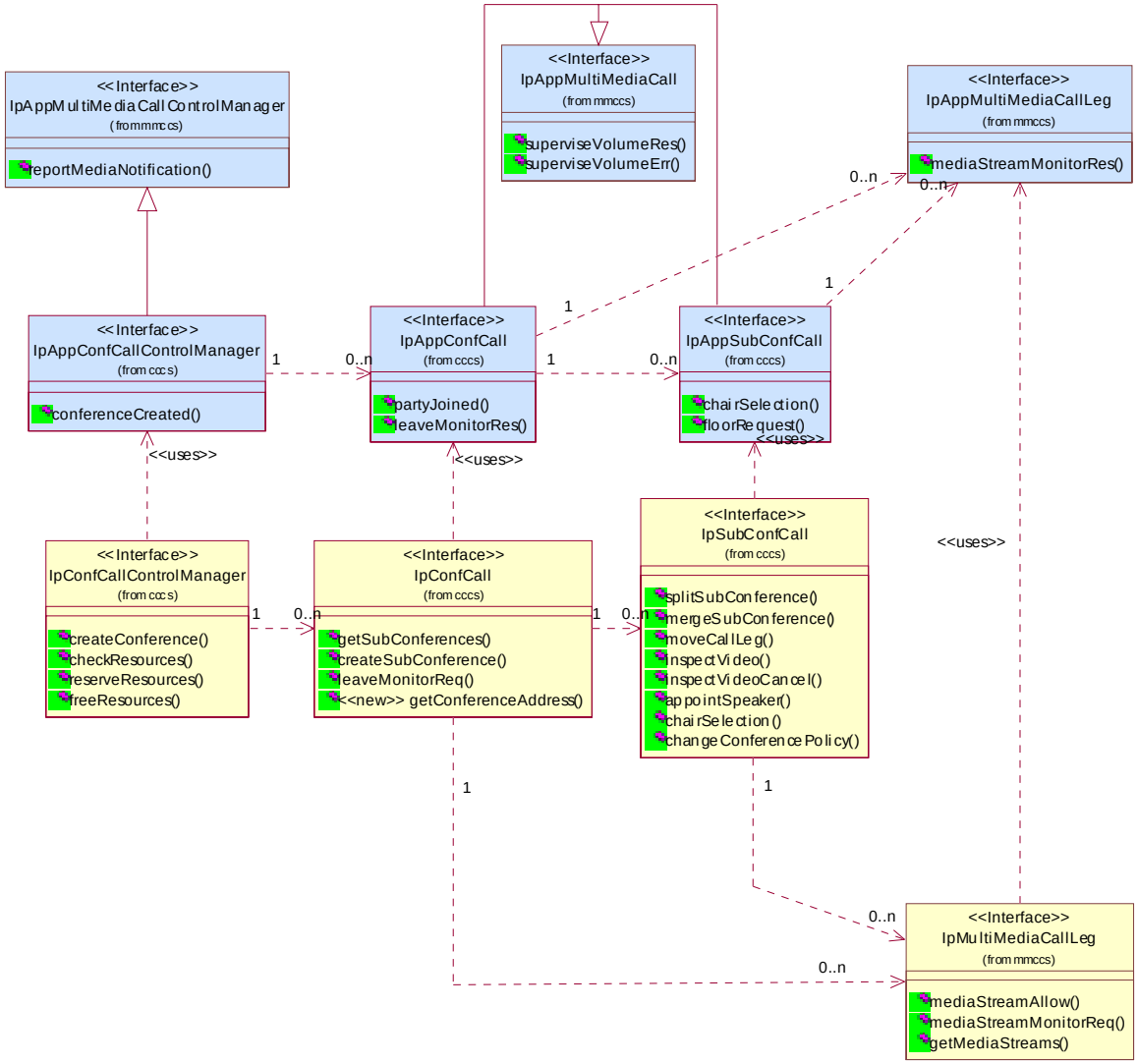


Figure 1: Application Interfaces

This class diagram shows the interfaces that make up the conference call control service package.

The diagram also shows the inheritance relation between the multi-party call interfaces and the conference call interfaces; the conference interfaces are specialisations of the corresponding multi-party call interfaces.

Furthermore, the class diagram illustrates that the conference call control manager can instantiate or be associated with zero or more conference calls. Each conference call can have one or more subconferences associated with it. Each subconference contains zero or more call legs associated. Detached legs are not associated with any specific subconference, instead they are associated with the conference call itself.

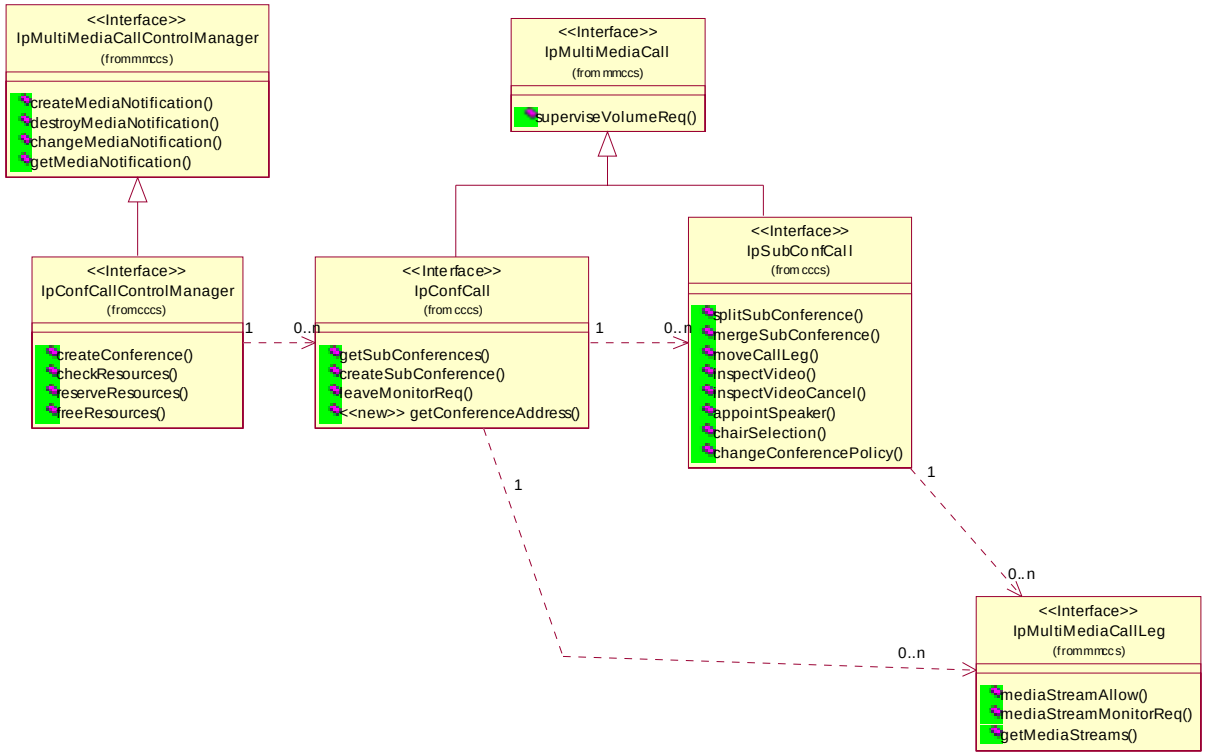


Figure 2: Service Interfaces

6 Conference Call Control Service Interface Classes

The Conference Call Control Service enhances the multi-media call control service. The conference call control service gives the application the ability to manipulate subconferences within a conference. A subconference defines the grouping of legs within the overall conference call. Only parties in the same subconference have a bearer connection (or media channel connection) to each other (e.g., can speak to each other). The application can:

- create new subconferences within the conference, either as an empty subconference or by splitting an existing subconference in two subconferences;

- move legs between subconferences;

- merge subconferences;

- get a list of all subconferences in the call;

The generic conference also gives the possibility to manipulate typical multi-media conference details, such as:

- interworking with network signalled conference protocols (e.g., H.323);

- manipulation of the media in the MCU, e.g., broadcasting of video;

- handling of multi-media conference policies, e.g., how video should be handled, voice controlled switched or chair controlled.

Furthermore the conference call control service adds support for the reservation of resources needed for conferencing. The application can:

- reserve resources for a predefined time period;

- free reserved resources;

- search for the availability of conference resources based on a number of criteria.

There are two ways to initiate a conference:

- the conferences can be started on the pre-arranged time by the service, at the start time indicated in the reservation. The application is notified about this. The application can then add parties to the conference and/or parties can dial-in to the conference using the address provided during reservation;

- the conference can be created directly on request of the application using the createConference method in the IpConfCallControlManager interface.

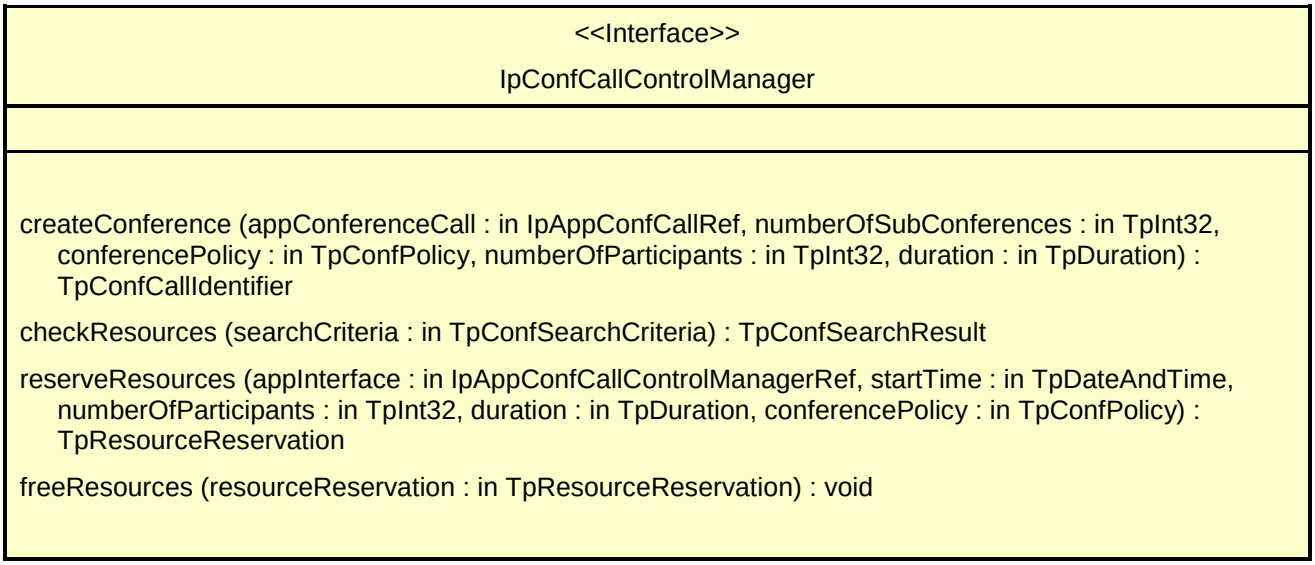
Each Conference Call Control interface inherits from a Multi Media Call Control interface, which in turn inherits from Multi Party Call Control. It is possible to implement conference call control without any multi-media features, using only those inherited methods which come from Multi Party Call Control, in addition to the Conference Call Control methods. The minimum required method set for each Conference Call Control interface reflects this possibility, by not requiring the Multi Media Call Control methods.

6.1 Interface Class IpConfCallControlManager

Inherits from: IpMultiMediaCallControlManager;

The conference Call Control Manager is the factory interface for creating conferences. Additionally it takes care of resource management.

This interface shall be implemented by a Conference Call Control SCF. As a minimum requirement, either the createConference() method shall be implemented, or the reserveResources() and freeResources() methods shall be implemented. The minimum required methods from IpMultiPartyCallControlManager are also required.



6.1.1 Method createConference()

This method is used to create a new conference. If the specified resources are not available for the indicated duration the creation is rejected with P_RESOURCES_UNAVAILBLE.

Returns conference : Specifies the interface reference and sessionID of the created conference.

Parameters

`appConferenceCall : in IpAppConfCallRef`

Specifies the callback interface for the conference created

`numberOfSubConferences : in TpInt32`

Specifies the number of subconferences that the user wants to create automatically. The references to the interfaces of the subconferences can later be requested with getSubConferences.

The number of subconferences should be at least 1.

`conferencePolicy : in TpConfPolicy`

Specifies the policy to be applied for the conference, e.g., are parties allowed to join (call into) the conference?

Note that if parties are allowed to join the conference, the application can expect partyJoined() messages on the IpAppConfCall interface.

`numberOfParticipants : in TpInt32`

Specifies the number of participants in the conference. The actual number of participants may exceed this, but these resources are not guaranteed, i.e., anything exceeding this will be best effort only and the conference service may drop or reject participants in order to fulfil other committed resource requests. By specifying 0, the application can request a best effort conference.

`duration : in TpDuration`

Specifies the duration for which the conference resources are reserved. The duration of the conference may exceed this, but after the duration, the resources are no longer guaranteed, i.e., parties may be dropped or rejected by the service in order to satisfy other committed resource requests. When the conference is released before the allocated duration, the reserved resources are released and can be used to satisfy other resource requests. By specifying 0, the application requests a best effort conference.

Returns

`TpConfCallIdentifier`

Raises

`TpCommonExceptions`

6.1.2 Method checkResources()

This method is used to check for the availability of conference resources.

The input is the search period (start and stop time and date) - mandatory.

Furthermore, a conference duration and number of participants can be specified - optional.

The search algorithm will search the specified period for availability of conference resources and tries to find an optimal solution.

When a match is found the actual number of available resources, the actual start and the actual duration for which these are available is returned. These values can exceed the requested values.

When no match is found a best effort is returned, still the actual start time, duration, number of resources are returned, but these values now indicate the best that the conference bridge can offer, e.g., one or more of these values will not reach the requested values.

Returns result : Specifies the result of the search. It indicates if a match was found. If no exact match was found the best attempt is returned.

Parameters

`searchCriteria : in TpConfSearchCriteria`

Specifies the boundary conditions of the search. E.g., the time period that should be searched, the number of participants.

Returns

TpConfSearchResult

Raises

TpCommonExceptions

6.1.3 Method reserveResources()

This method is used to reserve conference resources for a given time period. Conferences can be created without first reserving resources, but in that case no guarantees can be made.

Returns resourceReservation : Specifies a structured data type which contains two fields:

ResourceID: The address with which the conference can be addressed, both in the methods of the interface and in the network, i.e., if joinAllowed is TRUE, parties can use this address to join the conference.

If no match is found the ResourceID contains an empty address.

ReservationID: Specifies the reservation made. It should be unique in a particular resource.

Parameters

appInterface : in IpAppConfCallControlManagerRef

Specifies the callback interface to be used when the conference is created in the network. The application will receive the conferenceCreated message when a conference is created in the network.

startTime : in TpDateAndTime

Specifies the time at which the conference resources should be reserved, i.e., the start time of the conference.

numberOfParticipants : in TpInt32

Specifies the number of participants in the conference. The actual number of participants may exceed this, but these resources are not guaranteed, i.e., anything exceeding this will be best effort only and the conference service may drop or reject participants in order to fulfil other committed resource requests.

duration : in TpDuration

Specifies the duration for which the conference resources are reserved. The duration of the conference may exceed this, but after the duration, the resources are no longer guaranteed, i.e., parties may be dropped or rejected by the service in order to satisfy other committed resource requests. When the conference is released before the allocated duration, the reserved resources are released and can be used to satisfy other resource requests.

conferencePolicy : in TpConfPolicy

The policy to be applied for the conference, e.g., are parties allowed to join (call into) the conference? Note that if parties are allowed to join the conference, the application can expect partyJoined() messages on the appConfCall.

Returns

TpResourceReservation

Raises

TpCommonExceptions

6.1.4 Method freeResources()

This method can be used to cancel an earlier made reservation of conference resources.

This also means that no ConferenceCreated events will be received for this conference.

Parameters

resourceReservation: in TpResourceReservation

Specifies the ResourceID and the ReservationID that were received during the reservation.

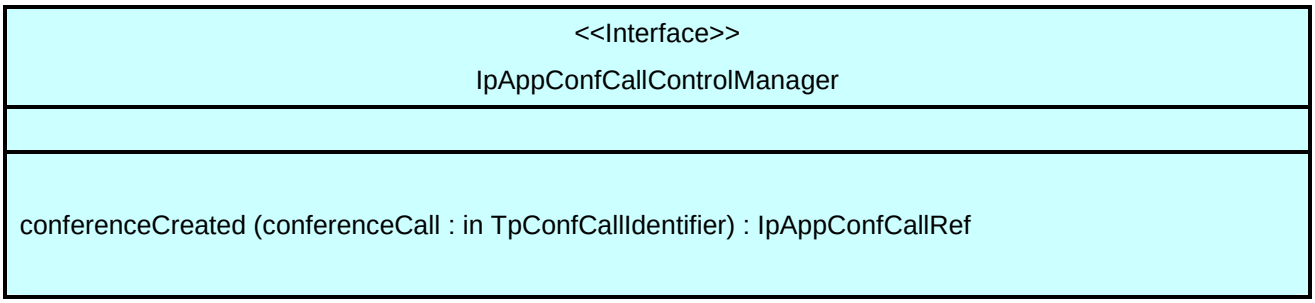
Raises

TpCommonExceptions

6.2 Interface Class IpAppConfCallControlManager

Inherits from: IpAppMultiMediaCallControlManager;

The conference call control manager application interface provides the application with additional callbacks when a conference is created by the network (based on an earlier reservation)



6.2.1 Method conferenceCreated()

This method is called when a conference is created from an earlier resource reservation.

Returns appInterface : Specifies a reference to the application interface which implements the callback interface for the new conference.

Parameters

conferenceCall: in TpConfCallIdentifier

Specifies the reference to the conference call interface to which the notification relates and the associated sessionID.

Returns

IpAppConfCallRef

6.3 Interface Class IpConfCall

Inherits from: IpMultiMediaCall;

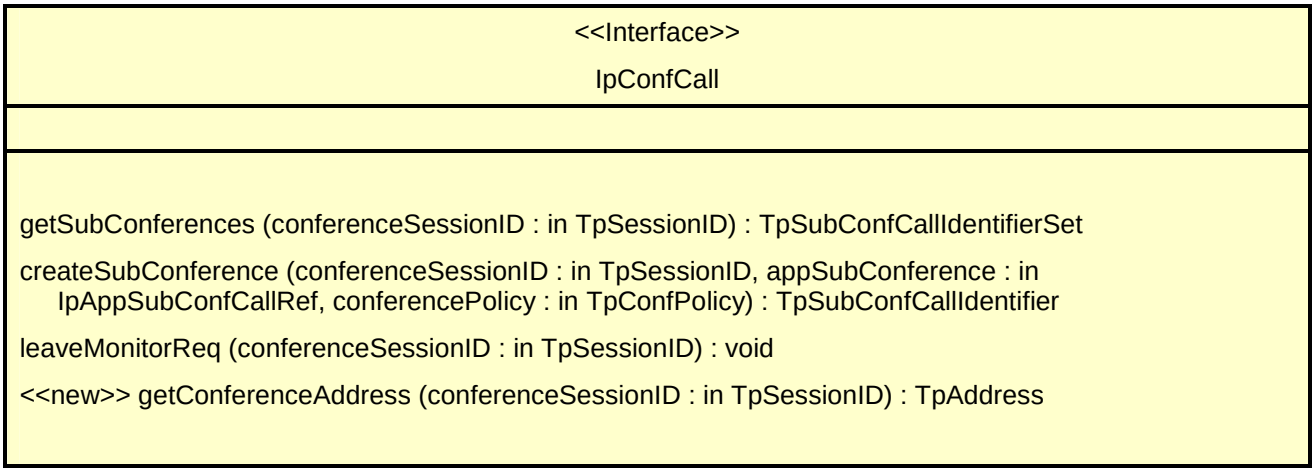
The conference call manages the subconferences. It also provides some convenience methods to hide the fact of multiple subconferences from the applications that do not need it. Note that the conference call always contains one subconference. The following inherited call methods apply to the conference as a whole, with the specified semantics:

- setCallback; changes the callback interface reference.
- release; releases the entire conference, including all the subconferences and detached legs.
- deassignCall; de-assigns the complete conference. No callbacks will be received by the application, either on the conference, or on any of the contained subconferences or call legs.
- getInfoReq; request information over the complete conference. The conference duration is defined as the time when the first party joined the conference until when the last party leaves the conference or the conference is released.

- setChargePlan; set the chargeplan for the conference. This chargeplan will apply to all the subconferences, unless another chargeplan is explicitly overridden on the subconference.
- superviseReq; supervise the duration of the complete conference.
- getCallLegs; return all the call legs used within the conference.
- superviseVolumeReq; supervises and sets a granted data volume for the conference.

Other methods apply to the default subconference. When using multiple subconferences, it is recommended that the application calls these methods directly on the subconference since this makes it more explicit what the effect of the method is:
- createAndRouteCallLegReq
- createCallLeg

This interface shall be implemented by a Conference Call Control SCF. As a minimum requirement, the getSubConferences(), getConferenceAddress() and createSubConference() methods shall be implemented. The minimum required methods from IpMultiPartyCall are also required.



6.3.1 Method getSubConferences()

This method returns all the subconferences of the conference.

Returns subConferenceList: Specifies the list of all the subconferences of the conference.

Parameters

`conferenceSessionID: in TpSessionID`

Specifies the sessionID of the conference.

Returns

TpSubConfCallIdentifierSet

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.3.2 Method createSubConference()

This method is used to create a new subconference. Note that one subconference is already created together with the conference.

Returns subConference: Specifies the created subconference (interface and sessionID).

Parameters

conferenceSessionID : in TpSessionID

Specifies the sessionID of the conference.

appSubConference : in IpAppSubConfCallRef

Specifies the call back interface for the created subconference.

conferencePolicy : in TpConfPolicy

Conference Policy to be used in the subconference. Optional; if undefined, the policy of the conference is used. Note that not all policy elements have to be applicable for subconferences.

Returns

TpSubConfCallIdentifier

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.3.3 Method leaveMonitorReq()

This method is used to request a notification when a party leaves the conference.

Parameters

conferenceSessionID : in TpSessionID

Specifies the session ID of the conference.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.3.4 Method <<new>> getConferenceAddress()

This method returns the conference address that specifies the address with which the conference can be addressed in case parties are allowed to join the conference.

Parameters

conferenceSessionID : in TpSessionID

Specifies the sessionID of the conference.

Returns

TpAddress

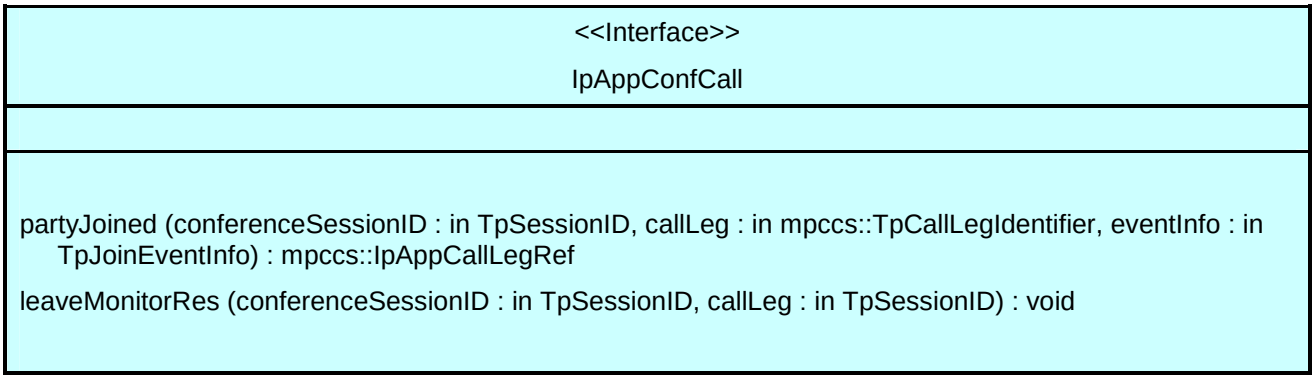
Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.4 Interface Class IpAppConfCall

Inherits from: IpAppMultiMediaCall;

The Conference Call application interface allows the application to handle call responses and state reports. Additionally it allows the application to handle parties entering and leaving the conference.



6.4.1 Method partyJoined()

This asynchronous method indicates that a new party (leg) has joined the conference. This can be used in, e.g., a meetme conference where the participants dial in to the conference using the address returned during reservation of the conference.

The Leg will be assigned to the default subconference object and will be in a detached state. The application may move the call Leg to a different subconference before attaching the media.

The method will only be called when joinAllowed is indicated in the conference policy.

Returns appCallLeg: Specifies the call back interface that should be used for callbacks from the new call Leg.

Parameters

conferenceSessionID : in TpSessionID

Specifies the session ID of the conference that the party wants to join.

callLeg : in mpccs::TpCallLegIdentifier

Specifies the interface and sessionID of the call leg that joined the conference.

eventInfo : in TpJoinEventInfo

Specifies the address information of the party that wants to join the conference.

Returns

mpccs::IpAppCallLegRef

6.4.2 Method leaveMonitorRes()

This asynchronous method indicates that a party (leg) has left the conference.

Parameters

conferenceSessionID : in TpSessionID

Specifies the session ID of the conference that the party wants to leaves.

callLeg : in TpSessionID

Specifies the sessionID of the call leg that left the conference.

6.5 Interface Class IpSubConfCall

Inherits from: IpMultiMediaCall;

The subconference is an additional grouping mechanism within a conference. Parties (legs) that are in the same subconference have a speech connection with each other. The following inherited call methods apply to the subconference as a whole, with the specified semantics:

- setCallback; changes the callback interface reference.
- release; releases the subconference, including all currently attached legs. When the last subconference in the conference is released, the conference is implicitly released as well.
- deassignCall; de-assigns the subconference. No callbacks will be received by the application on this subconference, nor will the gateway accept any methods on this subconference or accept any methods using the subconference as a parameter (e.g., merge). When the subconference is the last subconference in the conference, the conference is deassigned as well. In general it is recommended to only use deassignCall for the complete conference.
- getInfoReq; request information over the subconference. The subconference duration is defined as the time when the first party joined the subconference until when the last party leaves the subconference or the subconference is released.
- setChargePlan; set the charge plan for the subconference.
- superviseReq; supervise the duration of the subconference. It is recommended that this method is only used on the complete conference.
- superviseVolumeReq; supervises and sets a granted data volume for the subconference.
- getCallLegs; return all the call legs in the subconference.
- createCallLeg; create a call leg.
- createAndRouteCallLegReq; implicitly create a leg and route the leg to the specified destination.

This interface shall be implemented by a Conference Call Control SCF. As a minimum requirement, either the moveCallLeg() method shall be implemented, or the splitSubConference() and mergeSubConference() methods shall be implemented. The minimum required methods from IpMultiPartyCall are also required.

<<Interface>>

IpSubConfCall

splitSubConference (subConferenceSessionID : in TpSessionID, callLegList : in TpSessionIDSet, appSubConferenceCall : in IpAppSubConfCallRef) : TpSubConfCallIdentifier

mergeSubConference (subConferenceCallSessionID : in TpSessionID, targetSubConferenceCall : in TpSessionID) : void

moveCallLeg (subConferenceCallSessionID : in TpSessionID, targetSubConferenceCall : in TpSessionID, callLeg : in TpSessionID) : void

inspectVideo (subConferenceSessionID : in TpSessionID, inspectedCallLeg : in TpSessionID) : void

inspectVideoCancel (subConferenceSessionID : in TpSessionID) : void

appointSpeaker (subConferenceSessionID : in TpSessionID, speakerCallLeg : in TpSessionID) : void

chairSelection (subConferenceSessionID : in TpSessionID, chairCallLeg : in TpSessionID) : void

changeConferencePolicy (subConferenceSessionID : in TpSessionID, conferencePolicy : in TpConfPolicy) : void

6.5.1 Method splitSubConference()

This method is used to create a new subconference and move some of the legs to it.

Returns newSubConferenceCall: Specifies the new subconference that is implicitly created as a result of the method.

Parameters

subConferenceSessionID: in TpSessionID

Specifies the session ID of the subconference.

callLegList: in TpSessionIDSet

Specifies the sessionIDs of the legs that will be moved to the new subconference.

appSubConferenceCall: in IpAppSubConfCallRef

Specifies the application call back interface for the new subconference.

Returns

TpSubConfCallIdentifier

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.2 Method mergeSubConference()

This method is used to merge two subconferences, i.e., move all our legs from this subconference to the other subconference followed by a release of this subconference.

Parameters

subConferenceCallSessionID : in TpSessionID

Specifies the session ID of the subconference.

targetSubConferenceCall : in TpSessionID

The session ID of target subconference with which the current subconference will be merged.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.3 Method moveCallLeg()

This method moves one leg from this subconference to another subconference.

Parameters

subConferenceCallSessionID : in TpSessionID

Specifies the session ID of the source subconference.

targetSubConferenceCall : in TpSessionID

Specifies the sessionID of the target subconference.

callLeg : in TpSessionID

Specifies the sessionID of the call leg to be moved.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.4 Method inspectVideo()

This method can be used by the application to select which video should be sent to the party that is currently selected as the chair.

Whether this method can be used depends on the selected conference policy.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the multi media subconference.

inspectedCallLeg : in TpSessionID

Specifies the sessionID of call leg of the party whose video stream should be sent to the chair.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.5 Method inspectVideoCancel()

This method cancels a previous inspectVideo. The chair will receive the broadcasted video.

Whether this method can be used depends on the selected conference policy.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the multi media subconference.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.6 Method appointSpeaker()

This method indicates which of the participants in the conference has the floor. The video of the speaker will be broadcast to the other parties.

Whether this method can be used depends on the selected conference policy.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the multi media subconference.

speakerCallLeg : in TpSessionID

Specifies the sessionID of the call leg of the party whose video stream should be broadcast.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.7 Method chairSelection()

This method is used to indicate which participant in the conference is the chair. E.g., the terminal of this participant will be the destination of the video of the inspectVideo method.

Whether this method can be used depends on the selected conference policy.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the multi media subconference.

chairCallLeg : in TpSessionID

Specifies the sessionID of the call leg of the party that will become the chair.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.5.8 Method changeConferencePolicy()

This method can be used to change the conference policy in an ongoing conference.

Multi media conference policy options available. E.g.:

- chair controlled video / voice switched video;

- closed conference / open conference;

- Composite video (different types) / only speaker.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the multi media subconference.

conferencePolicy : in TpConfPolicy

New Conference Policy to be used in the subconference.

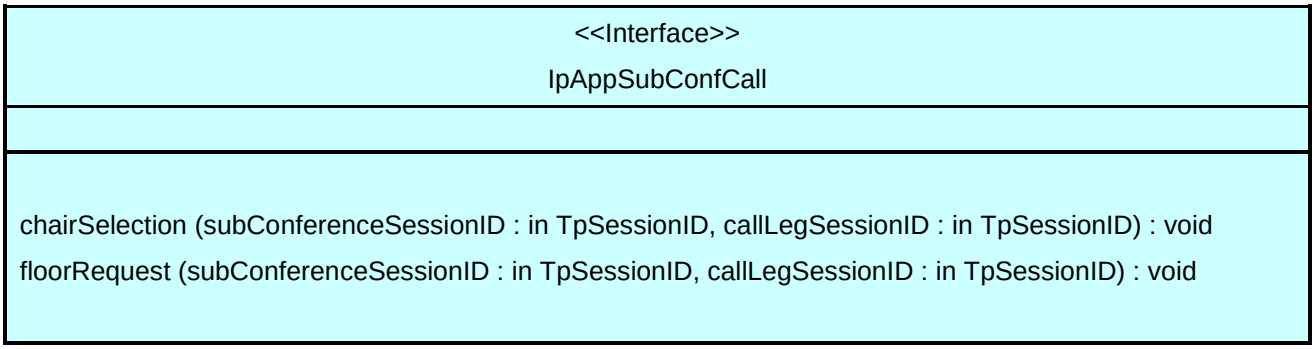
Raises

TpCommonExceptions, P_INVALID_SESSION_ID

6.6 Interface Class IpAppSubConfCall

Inherits from: IpAppMultiMediaCall;

The Sub Conference Call application interface allows the application to handle call responses and state reports from a sub conference.



6.6.1 Method chairSelection()

This method is used to inform the application about the chair selection requests from the network. The application can grant the request by calling the chairSelection method on the subconference.

Parameters

subConferenceSessionID : in TpSessionID

Specifies the session ID of the subconference where the chair request originates.

callLegSessionID : in TpSessionID

Specifies the session ID of the call leg making the chair request.

6.6.2 Method floorRequest()

This method is used to inform the application about the floor requests from the network. The application can grant the request by calling the appointSpeaker method.

Parameters

`subConferenceSessionID : in TpSessionID`

Specifies the session ID of the subconference where the floor request originates.

`callLegSessionID : in TpSessionID`

Specifies the session ID of the call leg making the floor request.

7 Conference Call Control Service State Transition Diagrams

There are no State Transition Diagrams for the Conference Call Control Service package.

8 Conference Call Control Data Definitions

This clause provides the Conference call control data definitions necessary to support the API specification.

The general format of a data definition specification is described below.

- Data Type

This shows the name of the data type.

- Description

This describes the data type.

- Tabular Specification

This specifies the data types and values of the data type.

- Example

If relevant, an example is shown to illustrate the data type.

All data types referenced in the present document but not defined in this clause are defined either in the common call control data definitions in ES 202 915-4-1 or in the common data definitions which may be found in ES 202 915-2.

8.1 Event Notification Data Definitions

No specific event notification data.

8.2 Conference Call Control Data Definitions

8.2.1 IpConfCall

Defines the address of an `IpConfCall` Interface.

8.2.2 IpConfCallRef

Defines a `Reference` to type IpConfCall.

8.2.3 IpAppConfCall

Defines the address of an `IpAppConfCall` Interface.

8.2.4 IpAppConfCallRef

Defines a `Reference` to type IpAppConfCall.

8.2.5 IpSubConfCall

Defines the address of an `IpSubConfCall` Interface.

8.2.6 IpSubConfCallRef

Defines a `Reference` to type IpSubConfCall.

8.2.7 IpAppSubConfCall

Defines the address of an `IpAppSubConfCall` Interface.

8.2.8 IpAppSubConfCallRef

Defines a `Reference` to type IpAppSubConfCall.

8.2.9 TpSubConfCallIdentifierSet

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

8.2.10 TpConfCallIdentifier

Defines the Sequence of Data Elements that unambiguously specify the Conference Call object

Sequence Element Name	Sequence Element Type	Sequence Element Description
ConfCallReference	IpConfCallRef	This element specifies the interface reference for the conference call object.
ConfCallSessionID	TpSessionID	This element specifies the session ID of the conference call.

8.2.11 TpSubConfCallIdentifier

Defines the Sequence of Data Elements that unambiguously specify the SubConference Call object

Sequence Element Name	Sequence Element Type	Sequence Element Description
SubConfCallReference	IpSubConfCallRef	This element specifies the interface reference for the subconference call object.
SubConfCallSessionID	TpSessionID	This element specifies the session ID of the subconference call.

8.2.12 IpAppConfCallControlManager

Defines the address of an `IpAppConfCallControlManager` Interface.

8.2.13 IpAppConfCallControlManagerRef

Defines a `Reference` to type IpAppConfCallControlManager.

8.2.14 TpConfPolicyType

Defines policy type for the conference.

If undefined the gateway will select an appropriate default.

If a mono media conference policy is specified for a multi-media conference, the gateway will select appropriate defaults for the multi-media policy items.

If a multi-media policy is selected for a mono-media (voice-only) conference, the multi-media conference items will be ignored.

Name	Value	Description
P_CONFERENCE_POLICY_UNDEFINED	0	Undefined
P_CONFERENCE_POLICY_MONOMEDIA	1	CCCS - monomedia conference policy
P_CONFERENCE_POLICY_MULTIMEDIA	2	MMCCS - multimedia conference policy

8.2.15 TpConfPolicy

Defines the `Tagged Choice of Data Elements` that specify the policy that needs adhered to by the conference.

	Tag Element Type	
	TpConfPolicyType	

Tag Element Value	Choice Element Type	Choice Element Name
P_CONFERENCE_POLICY_MONOMEDIA	TpMonoMediaConfPolicy	MonoMedia
P_CONFERENCE_POLICY_MULTIMEDIA	TpMultiMediaConfPolicy	MultiMedia

8.2.16 TpMonoMediaConfPolicy

Defines the type of conference policy as a sequence of Policy Items and their values.

For mono media there are only two types of conference policies; specified, i.e. the application provides the policy, or undefined, i.e. the GW may choose a default conference policy.

Sequence Element Name	Sequence Element Type	Description
JoinAllowed	TpBoolean	Specifies if dial-in to the conference is allowed. Parties can dial-in to the conference using the address returned during reservation. If this is specified the application will receive partyJoined for each participant dialling into the conference.

8.2.17 TpJoinEventInfo

Defines the `Sequence of Data Elements` that specify the information returned to the application in a Join event notification.

Sequence Element Name	Sequence Element Type
DestinationAddress	TpAddress
OriginatingAddress	TpAddress
OriginalDestinationAddress	TpAddress
RedirectingAddress	TpAddress
CallAppInfo	TpCallAppInfoSet

8.2.18 TpConfSearchCriteria

Defines the `Sequence of Data Elements` that specify the criteria for doing a search for available conference resources.

Sequence Element Name	Sequence Element Type
StartSearch	TpDateAndTime
StopSearch	TpDateAndTime
RequestedResources	TpInt32
RequestedDuration	TpDuration

8.2.19 TpConfSearchResult

Defines the `Sequence of Data Elements` that specifies the result of a search for available conference resources.

Sequence Element Name	Sequence Element Type
MatchFound	TpBoolean
ActualStartTime	TpDateAndTime
ActualResources	TpInt32
ActualDuration	TpDuration

8.2.20 TpMultiMediaConfPolicy

Sequence of items for multi-media conferences.

Sequence Element Name	Sequence Element Type	Description
JoinAllowed	TpBoolean	Specifies if dial-in to the conference is allowed. Parties can dial-in to the conference using the address returned during reservation. If this is specified the application will receive partyJoined for each participant dialling into the conference.
MediaAllowed	TpMediaType	Specifies the media that are allowed to be used by the participants. E.g. this can be used to limit the conference to audio only, even when all participants support video.
Chaired	TpBoolean	Specifies whether the conference is chaired or free. In a chaired conference the application or one of the participants acting as chair has special privileges; e.g. can control the video distribution.
VideoHandling	TpVideoHandlingType	Specifies how the video should be handled.

8.2.21 TpResourceReservation

Defines the `Sequence of Data Elements` that specifies the result of a search for available conference resources.

Sequence Element Name	Sequence Element Type	Sequence Element Description
ResourceID	TpAddress	The address with which the conference can be addressed
ReservationID	TpInt32	Specifies the reservation made. It should be unique in a particular resource

8.2.22 TpVideoHandlingType

Defines how video should be handled in the conference.

Name	Value	Description
P_MIXED_VIDEO	0	Video is mixed, no special treatment of speaker
P_SWITCHED_VIDEO_CHAIR_CONTROLLED	1	Video is switched, chair determines the speaker
P_SWITCHED_VIDEO_VOICE_CONTROLLED	2	Video is switched automatically based on audio output of the speaker

Annex A (normative): OMG IDL Description of Conference Call Control SCF

The OMG IDL representation of this interface specification is contained in a text file (cccs.idl contained in archive es_2029150405v010201p0.ZIP) which accompanies the present document.

Annex B (informative): W3C WSDL Description of Conference Call Control SCF

The W3C WSDL representation of this interface specification is contained in text files (cccs.wsdl contained in archive es_2029150405v010201p0.ZIP) which accompanies the present document.

Annex C (informative): Java API Description of the Call Control SCFs

The Java API representation of this interface specification can be obtained from the following URL:

- Java Call Control (http://jcp.org/jsr/detail/21.jsp)

Each JSR webpage contains a table identifying the relationships between the different versions of the Parlay, ETSI/OSA, 3GPP/OSA and JAIN SPA specifications. In addition, each JAIN SPA specification version indicates to which Parlay, ETSI/OSA and 3GPP/OSA specification versions it corresponds to.

Annex D (informative): Contents of 3GPP OSA Rel-5 Call Control

Conference Call Control does not form part of 3GPP Release 5 OSA specifications.

Annex E (informative): Record of changes

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

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

E.1 Interfaces

E.1.1 New

Identifier	Comments
Interfaces added in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Interfaces added in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.1.2 Deprecated

Identifier	Comments
Interfaces deprecated in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Interfaces deprecated in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.1.3 Removed

Identifier	Comments
Interfaces removed in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Interfaces removed in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.2 Methods

E.2.1 New

Identifier	Comments
Methods added in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
getConferenceAddress	
Methods added in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.2.2 Deprecated

Identifier	Comments
Methods deprecated in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Methods deprecated in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.2.3 Modified

Identifier	Comments
Methods modified in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Methods modified in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.2.4 Removed

Identifier	Comments
Methods removed in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Methods removed in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.3 Data Definitions

E.3.1 New

Identifier	Comments
Data Definitions added in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Data Definitions added in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.3.2 Modified

Identifier	Comments
Data Definitions modified in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Data Definitions modified in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.3.3 Removed

Identifier	Comments
Data Definitions removed in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Data Definitions removed in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.4 Service Properties

E.4.1 New

Identifier	Comments
Service Properties added in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Service Properties added in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.4.2 Deprecated

Identifier	Comments
Service Properties deprecated in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Service Properties deprecated in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.4.3 Modified

Identifier	Comments
Service Properties modified in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Service Properties modified in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.4.4 Removed

Identifier	Comments
Service Properties removed in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Service Properties removed in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.5 Exceptions

E.5.1 New

Identifier	Comments
Exceptions added in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Exceptions added in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.5.2 Modified

Identifier	Comments
Exceptions modified in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Exceptions modified in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.5.3 Removed

Identifier	Comments
Exceptions removed in ES 202 915-4-5 version 1.1.1 (Parlay 4.0)	
Exceptions removed in ES 202 915-4-5 version 1.2.1 (Parlay 4.1)	

E.6 Others

History

Document history		
V1.1.1	January 2003	Publication
V1.2.1	June 2003	Membership Approval Procedure MV 20030801: 2003-06-03 to 2003-08-01
V1.2.1	August 2003	Publication