Final draft ETSI ES 204 915-4-1 V1.1.1 (2008-02)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 1: Call Control Common Definitions (Parlay 6)

Reference
DES/TISPAN-01032-4-01-OSA

Keywords
API, IDL, OSA, UML

ETSI

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

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

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

Important notice

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

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

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

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

Contents

Intellectual Property Rights 5
Foreword 5
1 Scope 7
2 References 7
3 Definitions and abbreviations 7
3.1 Definitions 7
3.2 Abbreviations 7
4 Call Control SCF 7
4.1 Call Model Description 8
4.2 Structure of Call Control SCF Documentation 8
4.3 General requirements on support of methods 9
4.4 Application Control of a Call or Session 9
4.4.1 Introduction 9
4.4.2 Concept of Multiple Points of Control 9
4.4.2.1 Cascaded Chain Model 9
4.4.3 Service Interactions 11
4.4.4 Multiple services - applicability of MPC for Parlay/OSA 11
5 The Service Interface Specifications 11
5.1 Interface Specification Format 11
5.1.1 Interface Class 11
5.1.2 Method descriptions 12
5.1.3 Parameter descriptions 12
5.1.4 State Model 12
5.2 Base Interface 12
5.2.1 Interface Class IpInterface 12
5.3 Service Interfaces 12
5.3.1 Overview 12
5.4 Generic Service Interface 13
5.4.1 Interface Class IpService 13
5.4.1.1 Method setCallback() 13
5.4.1.2 Method setCallbackWithSessionID() 13
6 Common Call Control Data Types 14
6.1 TpCallAlertingMechanism 14
6.2 TpCallBearerService 14
6.3 TpCallChargePlan 14
6.4 TpCallPartyToChargeAdditionalInfo 15
6.5 TpCallPartyToChargeType 15
6.6 TpCallChargeOrderCategory 15
6.7 TpCallEndedReport 15
6.8 TpCallError 15
6.9 TpCallAdditionalErrorInfo 16
6.10 TpCallErrorType 16
6.11 TpCallInfoReport 16
6.12 TpCallInfoType 17
6.13 TpCallLoadControlMechanism 17
6.14 TpCallLoadControlIntervalRate 17
6.15 TpCallLoadControlMechanismType 17
6.16 TpCallMonitorMode 17
6.17 TpCallNetworkAccessType 18
6.18 TpCallPartyCategory 18
6.19 TpCallServiceCode 18
6.20 TpCallServiceCodeSet 18
6.21 TpCallServiceCodeType 19

6.22 TpCallSuperviseReport 19
6.23 TpCallSuperviseTreatment 19
6.24 TpCallTeleService 20
6.25 TpCallTreatment 20
6.26 TpCallTreatmentType 20
6.27 TpCallAdditionalTreatmentInfo 21
6.28 TpMediaType 21

Annex A (normative): OMG IDL Description of Common Call Control Data Types 22

Annex B (informative): W3C WSDL Description of Common Call Control Data Types 23

Annex C (informative): Java API Description of the Call Control SCFs 24

Annex D (informative): Contents of 3GPP OSA Rel-7 Call Control 25

Annex E (informative): Description of Call Control Sub-part 1: Call Control Common Definitions for 3GPP2 cdma2000 networks 26

E.1 General Exceptions 26

E.2 Specific Exceptions 26
E.2.1 Clause 1: Scope 26
E.2.2 Clause 2: References 26
E.2.3 Clause 3: Definitions and abbreviations 26
E.2.4 Clause 4: Call Control SCF 26
E.2.5 Clause 5: The Service Interface Specifications 26
E.2.6 Clause 6: Common Call Control Data Types 26
E.2.7 Annex A (normative): OMG IDL Description of Common Call Control Data Types 27
E.2.8 Annex B (informative): W3C WSDL Description of Common Call Control Data Types 27
E.2.9 Annex C (informative): Java™ API Description of the Call Control SCFs 27

Annex F (informative): Record of changes 28

F.1 Interfaces 28
F.1.1 New 28
F.1.2 Deprecated 28
F.1.3 Removed 28

F.2 Methods 28
F.2.1 New 28
F.2.2 Deprecated 28
F.2.3 Modified 29
F.2.4 Removed 29

F.3 Data Definitions 29
F.3.1 New 29
F.3.2 Modified 29
F.3.3 Removed 29

F.4 Service Properties 29
F.4.1 New 29
F.4.2 Deprecated 29
F.4.3 Modified 30
F.4.4 Removed 30

F.5 Exceptions 30
F.5.1 New 30
F.5.2 Modified 30
F.5.3 Removed 30

F.6 Others 30

History 31

Intellectual Property Rights

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

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

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 4, sub-part 1 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 204 915) 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";

Part 15: "Multi-Media Messaging SCF"

Part 16: "Service Broker 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 6.0 set of specifications.

The present document is equivalent to 3GPP TS 29.198-4-1 V7.0.0 (Release 7).

1 Scope

The present document is part 4, sub-part 1 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 common definitions used by the Call Control Service Capability Features (SCF).

2 References

The references listed in clause 2 of ES 204 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 204 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 6)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 204 915-1 apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ES 204 915-1 apply.

4 Call Control SCF

Four flavours of call control APIs are specified in ES 204 915, Parlay 6. Three of these have been included in 3GPP Release 6. These are the Generic Call Control (GCC), the Multi-Party Call Control (MPCC) and the Multi-Media Call Control (MMCC). The generic call control is the same API as was already present in the previous specification for 3GPP Release 99 (TS 129 198 V3). Multi-Party Call Control was introduced in the Release 4 specifications, and Multi-Media Call Control was introduced in Release 5. All four have been included in ES 201 915, Parlay 3 and ES 202 915, Parlay 4.

The joint work between 3GPP CN5, ETSI TISPAN and the Parlay Call Control Working group with collaboration from JAIN has been focussed on the Multi-party and Multi-Media call control APIs. A number of improvements on call control functionality have been made and are reflected in these APIs. For this it was necessary to break the inheritance that previously existed between Generic and Multi-party call control.

The joint call control group has furthermore decided that the multi-party call control is to be considered as the future base call control family and the technical work will not be continued on Generic Call control. Errors or technical flaws will of course be corrected.

4.1 Call Model Description

The call model used for the Call Control SCFs has the following objects:

- A call object. A call is a relation between a number of parties. The call object relates to the entire call view from the application. E.g. the entire call will be released when a release is called on the call. Note that different applications can have different views on the same physical call, e.g. one application for the originating side and another application for the terminating side. The applications will not be aware of each other, all 'communication' between the applications will be by means of network signalling. The API currently does not specify any feature interaction mechanisms.
- A call leg object. The leg object represents a logical association between a call and an address. The relationship includes at least the signalling relation with the party. The relation with the address is only made when the leg is routed. Before that the leg object is IDLE and not yet associated with the address.
- An address. The address logically represents a party in the call.
- A terminal. A terminal is the end-point of the signalling and/or media for a party. This object type is currently not addressed.

The call object is used to establish a relation between a number of parties by creating a leg for each party within the call.

Associated with the signalling relationship represented by the call leg, there may also be a bearer connection (e.g. in the traditional voice only networks) or a number (zero or more) of media channels (in multi-media networks).

A leg can be attached to the call or detached from the call. When the leg is attached, this means that media or bearer channels related to the legs are connected to the media or bearer channels of the other legs that are attached to the same call. I.e. only legs that are attached can 'speak' to each other. A leg can have a number of states, depending on the signalling received from or sent to the party associated with the leg. Usually there is a limit to the number of legs that are in being routed (i.e. the connection is being established) or connected to the call (i.e. the connection is established). Also, there usually is a limit to the number of legs that can be simultaneously attached to the same call.

Some networks distinguish between controlling and passive legs. By definition the call will be released when the controlling leg is released. All other legs are called passive legs. There can be at most one controlling leg per call. However, there is currently no way the application can influence whether a Leg is controlling or not.

There are two ways for an application to get the control of a call. The application can request to be notified of calls that meet certain criteria. When a call occurs in the network that meets these criteria, the application is notified and can control the call. Some legs will already be associated with the call in this case. Another way is to create a new call from the application.

4.2 Structure of Call Control SCF Documentation

Each of the Call Control SCFs is specified under the following headings:

- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause shows how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data definitions clause shows a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 204 915-2.

4.3 General requirements on support of methods

An implementation of one of the call control APIs which supports or implements a method described in one of the sub-parts of ES 204 915-4, 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.

4.4 Application Control of a Call or Session

4.4.1 Introduction

Services should be provision-able by multiple independent parties and therefore multiple applications may apply control to the same instance of a call or a session. How this may be enabled is further described in the following.

However, first some reflections on what is meant with application control:

- **Single application control** may be classified as to allow at the same point in time during call or session processing only one application to be capable to influence the call or session. This does not exclude more applications on the same call, but they cannot operate at the same time. This is referred to as "***Single point of Control (SPC)***" in IN terminology.

- **Multiple application control** may be classified as to allow at the same point in time during call or session processing more than one application to be capable to influence the call or session. This is referred to as ***"Multiple Points of Control (MPC)"*** in IN terminology.
  MPC will demand some rules for event handling among multiple applications on a call like the cascaded chain principle as applied in IN CS3 (ITU-T Recommendation Q.1238-2), where MPC has been introduced.

4.4.2 Concept of Multiple Points of Control

The term "multiple points of control" refers to the situation when multiple concurrently executing applications apply control to one and the same instance of a call or session.

General Objective:

"If there ismore than one controlling application acting on the same call or session, then the event notification detection point processing requested by any of the involved applications shall be performed in the same way as if notification reporting had occurred in different call or session control instances, which are separated by a Network Node interface".

NOTE: The objective description above is taken from ITU-T Recommendation Q.1238-2, but slightly generalized to become non-IN specific.

The MPC general objective signifies the cascaded chain principle, which is further explained below through an informative cascaded chain model.

4.4.2.1 Cascaded Chain Model

When services running in different nodes apply control to a call or session, even if they are unaware of each other, they provide a cascading of applications. They provide a natural ordering of applications. This ordering can be said to be upstream or downstream. A downstream ordering is the ordering of applications as they are invoked downstream in the network from the calling party (e.g. origin client, UAC) toward the called party (e.g. destination server, UAS). An upstream ordering is the ordering of applications as they are invoked upstream in the network from the called party toward the calling party.

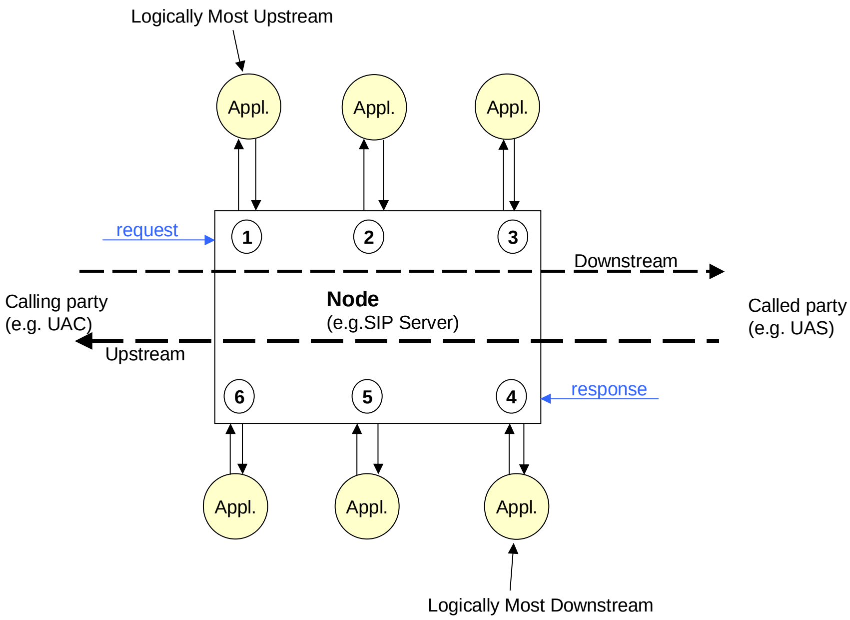

Figure 1: Cascaded Chain Model

When applications that apply control to one instance of a call or session are invoked in some order, on the same node, the applications can be thought of as cascaded.

When a "request" is received then the earlier the application is invoked, based on this event, the more logically upstream it is considered to be in the chain of cascaded applications. When a response is received then the earlier the applications is invoked, based on this event, the more logically downstream it is considered to be in the chain of cascaded applications.

This means that the applications are treated as if they were triggered in different nodes (or hosts). This model is conceptually simple and may provide a natural algorithm for resolving conflicts between the instructions of multiple service applications at the same call or session event.

On reception of a call or session event in the network the actions are executed in order of priority in the following manner:

1) Control is passed to the first application.

2) Some response is received from the first application.

3) Control is passed to the second application.

4) Some response is received from the second application and so on.

In this way a decision about whether to invoke a subsequent application can depend on the output from the previous application.

If the first application terminates the request then the second application must not be invoked.

The most simple form of cascading is an order based in which the applications are triggered on a single event.

If for example three application "X1" "Y1" and "Z1" need to be triggered on event "e"; then specifying an order say X1-Y1-Z1 for event e means that first "X1" is contacted and its instructions taken, then the output of this is passed to "Y1" and finally the output of "Y1" is passed to "Z1". (The letter represents the application and the number an instance of the application running for a given user.)

How about future events that should invoke the same instances of "X" "Y" and "Z"? For these events there could be a totally different order, specified say YXZ. It would however be administratively very complex if one has to specify an order for all instances for all events. Thus a general basic principle may be useful like the cascaded chain for defining a default ordering in the cases where this is not possible. The actual order depends on whether the event is coming from the calling party (downstream) or the called party (upstream).

If however a new application A1 should be triggered on "er" then it would be necessary to place this somewhere in the order. However, a default general assumption can be that event reporting to already invoked applications should have precedence over the invocation of new applications (see ITU-T Recommendation Q.1238-2).

Anyhow, as a network operator's option any MPC generalized rules specified may be replaced or enhanced by network operator specific rules.

4.4.3 Service Interactions

A variety of different services, service enablers, and capabilities are being standardized. There are potential feature interactions among these various services, service enablers, and capabilities.

Conflicts, incompatibilities, or modifications of one feature's characteristics due to interactions with another active feature need to be resolved in case of multiple services.

In case there are multiple initial notification requests (filter criteria) assigned for one subscriber, a priority describe the order in which the applications shall be contacted when the call or session encounters an event that matches the initial filter criteria. Handling of service/application interaction issues to prevent undesired interactions when multi services are to be supported is outside the scope of the present document. It is the basic assumption that the network operator is responsible for the provisioning of triggers in the network as in this domain full awareness exists of all other services and applications.

4.4.4 Multiple services - applicability of MPC for Parlay/OSA

As far as the OSA/Parlay API is concerned a user can choose his clients as he wishes, i.e. the user may subscribe to more services over the network, each one being to a separate client application, not necessary placed in the same physical entity.

From an OSA/Parlay gateway side it is a network implementation issue if multiple or single point of control mechanisms are supported. Multi service support extends the number of applications that can register for notifications. An example of multi service support network could be an OSA/Parlay Gateway in a UMTS IMS network or IN network complying with the multiple points of control principle as defined for IN CS3.

Overlapping criteria have been defined for GCCS and MPCCS to prevent multiple points of control, leading to possible interaction problems. Where Multi service support is provided, the overlap criteria rules as defined to secure single point of control can be overruled.

5 The Service Interface Specifications

5.1 Interface Specification Format

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

5.1.1 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>.