ETSI EG 201 988-2 V1.1.1 (2003-04)

ETSI Guide

**Services and Protocols for Advanced Networks (SPAN);
Service Provider Access Requirements (SPAR);
Open Service Access for API requirements;
Part 2: Version 2**

Reference

DEG/SPAN-141606-2

Keywords

API, architecture, interface, UML

Contents

Intellectual Property Rights 4

Foreword....4

Introduction 4

1 Scope 5

2 References 5

3 Abbreviations 5

4 ETSI OSA Phase2/Parlay Phase 4 API domains....6
4.1 Framework interface and service interface....6

5 Proposed enhancements to existing interfaces 6
5.1 General requirements 6
5.1.1 Backwards compatibility/deprecation....6
5.1.2 Emergency preparedness 7
5.1.3 Balancing up of interfaces 7
5.2 Framework 8
5.2.1 Framework information model 8
5.2.2 Framework management tool 9
5.2.3 Enhancements on event notification handling 10
5.2.4 Framework operator administration interfaces 11
5.3 Call Control....12
5.3.1 IM Session control functions 12
5.3.2 Packet switching Call Control functions....13
5.4 Terminal capabilities....13
5.4.1 Discovery of client terminal capabilities 13
5.5 User interaction 14
5.5.1 Interact with a user....14
5.6 Charging issues related to Call Control....14
5.6.1 GCC....14
5.6.2 Multi media call control....14
5.7 Event notification 15
5.8 Network controlled notifications....15
5.9 Content based charging 16
5.9.1 Service properties 16
5.9.2 User confirmation 17
5.9.3 Support of roaming/Multi-network scenarios 18
5.9.4 Separation of rating and non-rating functionality 18
5.9.5 Split charging....19

6 New interfaces and areas of involvement....19
6.1 Information transfer function 19
6.2 Presence related capability functions 20
6.3 Policy management 20
6.4 Parlay and SIP 21
6.5 Inclusion of SOAP/XML as an alternative transport mechanism....21

Annex A (informative): Bibliography....23

History 24

Intellectual Property Rights

Foreword

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

The present document is part 2 of a multi-part deliverable covering Service Provider Access Requirements (SPAR); Open Service Access for API requirements, as identified below:

Part 1: "Version 1";

Part 2: "Version 2";

Part 3: "Version 3".

Introduction

The present document contains the Requirements capture for ETSI Version 2 Open Service Access API protocol specification.

1 Scope

The present document contains the functional requirements for the second phase of the ETSI Open Service Access API, as defined in ES 202 915 [4]. The present document has been compiled in conjunction with Parlay and represents the fourth phase of the Parlay API. The ETSI and Parlay API has been specified and designed using the requirements identified in the present document. The requirements are intended to provide the necessary functionality for benchmark applications.

The new requirements build upon the ETSI OSA Phase 1 API in ES 201 915 [3] and the Parlay 3 specification and should be fully backward compatible. This means that any network operator implementing ETSI OSA Phase 2 or Parlay 4 should be able to interwork with a client application provider implementing ETSI OSA Phase 1 or Parlay 3. In other words ETSI OSA Phase 2 and Parlay 4 will retain ETSI OSA Phase 1 and Parlay 3 as a complete subset.

2 References

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

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI TS 129 198: "Universal Mobile Telecommunications System (UMTS); Open Service Access (OSA) Application Programming Interface (API)".

[2] ETSI TS 123 127: "Universal Mobile Telecommunications System (UMTS); Virtual Home Environment (VHE) / Open Service Access (OSA); Stage 2 (3GPP TS 23.127 version 5.2.0 Release 5)".

[3] ETSI ES 201 915: "Open Service Access (OSA); Application Programming Interface (API)".

[4] ETSI ES 202 915: "Open Service Access (OSA); Application Programming Interface (API)".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Program Interface
ASP	Advanced Signal Processor
CC	Call Control
DSC	Data Switch Control
ETS	Emergency Telecommunications Service
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
LIF	Location Interoperability Forum
MIS	Management Information System
MPCC	Multi Protocol Call Control
OSA	Open Services Access
PDP	Packet data protocol
SIP	Session Initiation Protocol
SCS	Service Capability Server
SOAP	Simple Object Access Protocol

SS7	Signalling System 7
USSD	Unstructured SS Data
W3C	World Wide Web Consortium
XML	eXtended Markup Language

4 ETSI OSA Phase2/Parlay Phase 4 API domains

The ETSI/Parlay API is an open, technology-independent, and extensible interface into networking technologies. The API is therefore applicable to a number of business and application domains, not just telecommunications network operators.

Examples of business domains that may use the API include:

- Third Party Telephony Service Providers.
- Interactive multi-media Service Providers.
- Corporate Businesses.
- Small Businesses.
- Residential Customers.
- Network Operators.

All of these businesses have networking requirements, ranging from simple telephony and call routing to call centres, virtual private networks and fully interactive multi-media.

The rest of the present document is structured to capture all of the requirements that are deemed necessary to enhance the existing ETSI OSA Phase 1 and Parlay 3.2 specification to an ETSI OSA Phase 2/Parlay 4.0 status.

4.1 Framework interface and service interface

The API provides the common interfaces to a variety of services. For the services to work together in a coherent fashion, "framework" functions are required and are also included in the present document.

Services and the framework functionality will be exposed via interfaces. These interfaces will be called the service interface and framework interface respectively.

5 Proposed enhancements to existing interfaces

5.1 General requirements

5.1.1 Backwards compatibility/deprecation

Source: Parlay

Issue/Motivation:

It needs to be considered what can be done if we find that certain interfaces in Parlay 3.1 are found to be unstable and therefore require appropriate modification. If we use the concept of deprecation then we can effectively provide new methods to that interface where the old methods are incorrect. This means that the two methods will exist side by side in the same interface for the same purpose in Parlay 4.0. One method may not be complete but the other is! The methods that are incorrect would be removed in further versions of the API.

Requirement description:

The Parlay 4.0/OSA 2.0/ETSI SPAN 2.0 APIs shall be backwards compatible. This has two aspects:

- A client application utilizing Parlay 3.0/OSA 1.0/ETSI SPAN 1.0 APIs shall run without change (not even re-compilation) against a server providing Parlay 4.0/OSA 2.0/ETSI SPAN 2.0 APIs.
- A deprecation mechanism shall be defined that allows to remove outdated methods or interfaces in a well-defined, step-wise approach.

5.1.2 Emergency preparedness

Source: Telcordia

Issue/Motivation:

There is a need to extend A/IN-based facilities defined for national emergency calls to Next Generation Networks and APIs. The U.S. Government and other countries have sponsored programs over the past 15 years to ensure, via standards and implementation programs, that National Emergency calls enjoy priority handling, Network Management Control exception and Alternate Carrier Routing, etc. This initiative is known as Emergency Telecommunications Service (ETS). Although this is currently available for voice calls only there is also a need to handle new types of communication (data, email, video, multi-media), new types of networks (wireless, packet) and technology (protocols, architectures). These requirements impact Call Control and Policy Management and may also impact Mobility, Charging and Framework (e.g. for location-based service, accounting bypass, security, etc.).

Requirement description:

The Parlay/OSA/ETSI SPAN APIs shall support the Emergency Telecommunications Service. In particular, they shall provide client applications with means to make use of the Emergency Telephony Service.

Possible solution and further considerations:

To support the Emergency Telecommunications Service, an optional parameter could be provided within the call control and other interfaces (e.g. CC, MPCC, DSC).

5.1.3 Balancing up of interfaces

Source: Eurescom

Issue/Motivation:

Many of the Parlay/OSA/ETSI SPAN APIs are highly asymmetric between application and gateway. As the capabilities of terminals connected to networks continues to grow, network based applications will require greater awareness of these capabilities. Likewise, as new network protocols such as SIP, which are peer-to-peer in nature, are developed, new types of functionality and control will become possible. The "Balancing up" of interfaces is concerned with identifying areas where the asymmetry of Parlay may cause limitation in functionality or feature interaction problems.

Although many of these asymmetries are particularly apparent on SIP networks, many aspects identified will not be restricted to SIP.

The work is aimed at identifying and suggesting changes to Parlay to help in solving feature interaction problems and to support fully the flexibility which new networks, protocols and terminals enable.

Scenarios:

This clause provides scenarios and examples of situations where more balanced interfaces might be beneficial.

At present the work is expected to have a broad impact across the service interfaces, for example.

Call Control

An application can create a new call leg within a call, but a network cannot create a call leg and notify an application that a call leg has been created. Terminals and protocols which allow a client to add an additional party to the call should be able to do so, making the application aware of the new party.

An application can attach and detach call legs from a call, but the application cannot be notified that the call legs have been attached or detached by the connected party. This causes problems because if a caller detaches from a call, then the application is not aware of this. In a symmetrical model, the media stream could be connected or disconnected by either application or client, and the application could be notified of changes.

Mobility interfaces

The mobility interfaces allow an application to query the status and location of an address. However, an application cannot request notification of incoming requests for the status or location of address and return a response. A scenario where this causes a problem is where a unified communications application provides a "one-number" service, and a second application requests the status or location of the user. The unified communications application must have be able to receive status requests and respond to them.

Requirements description:

- To identify aspects of Parlay where asymmetries can cause limitations. This includes call control, mobility, terminal capabilities and user interaction interfaces.
- To collate the information from the comparisons and to identify service scenarios where the existing interfaces are too restrictive and where symmetrical behaviour would be advantageous.
- To ensure that any symmetrical interfaces proposed do not compromise compatibility with existing asymmetric networks.

Possible solution and further consideration:

Modified Parlay interface class diagrams, interface definitions and data types or recommendations via reports.

5.2 Framework

5.2.1 Framework information model

Source: Eurescom

Issue/Motivation:

An Information Model for the Parlay/OSA Framework (FIM, Framework Information Model) can answer to several needs, coming from different actors in the possible business models, but the first reason to define such a model is to have an agreed, common view and a "common language" to address the same concepts when analysis related to complex data structures, relationships and objects, have to be done. Since work done so far, both on standard and on the vendors' side, indicates that the Framework functionality is the heart of the Parlay Gateway system, the availability of a Framework Information Model appears more and more relevant.

The needs of new interfaces (e.g. user profile service interface), or to improve the existing ones, makes the FIM definition relevant also on the standard side. In particular, a clear definition of the objects belonging to the framework domain, and how they interact, can help and shorten the analysis/modelling phase of possible new Parlay services; consequently the definition of new interfaces and improvements/enhancements of existing ones can take advantage of a FIM in terms of speed, easiness, completeness etc.

Vendors planning to develop Parlay Gateways have surely to face the problem of defining data/information models of their system, including a model of the framework functionality, but such models will be obviously tied to the specific implementation. In addition, parts of a FIM are already, implicitly, defined in the specification of current framework APIs, such as service subscription, service registration, and service properties.

The Framework Information Model, objective of the present work, to be useful in different contexts, should be carefully structured and defined at a proper detail level: not so deep to impose implementation constraints and not so abstract to hide aspects (namely entities, relationships, objects or other) that are needed to achieve the aforesaid objectives.

Scenarios:

This clause describes two possible scenarios/contexts that could take advantage of the FIM.

The Service Registration is a good example of a scenario that could take advantage in using a Framework model. Before a service can be accessed and used, it has to be registered in the Framework, where information on the available services is contained. The registration procedure is described in the specifications, as well as some relationships among the involved entities (namely Service Supplier Administrator, Framework, the Service that has to be registered); moreover some of the involved objects can be deduced from the interface definition, but a clear, high level view of the whole procedure is not available. Part of this issue could be covered by FIM.

The second scenario a FIM can be useful, is related to Service Subscription (this subject is treated in clause 4.17 of "Framework Interfaces, Client Application View – Version 2.1" specification). Even though the Service Subscription phase is described by the specification in some detail (e.g. in terms of a "Subscription Business Model", defining high level actors and relationships, with pictures and text), it is not simple to analyse all the relationships among the various involved entities (e.g. the Enterprise Operator, the Framework and the Client Application, that have precise roles in the Service Subscription phase), or to have a clear vision, as an example, of all the SAG (Subscription Assignment Group) aspects. A more formal and detailed representation, including the involved Framework entities that can be defined in the Framework model, can be helpful.

Requirements description:

A useful Framework Information Model should be able to support analysis (as mentioned before both standard-oriented and implementation oriented) of several aspects. In particular the following functionalities should be addressed:

1) Service Subscription and subscription management: objects and relationships description.

2) Service Interface and Framework Interface subscription by applications: each involved entity should be correctly defined, together with the configuration data related to application subscription. Possible aspects related SLA constraints to applications subscription should be addressed as well.

3) Services and service interfaces registration and configuration.

4) Service discovery.

5) Usage data management (e.g. logging data, Service Interface usage data).

Possible solution and further consideration:

Services and Service Interfaces configurations are typically done through properties. A property is composed of a name, a type and a value (data or policy). Moreover, each property can be tied to validation rules. The model should be able to cover these aspects (and consider their relationship with service properties).

A further need is that the model should be easily extensible, to allow Service Interfaces incremental introduction. Each Service Interface is characterized by set of properties. Some of such properties can be defined by standards, others can be proprietary, marking out a particular implementation.

The FIM definition will consist of a formal UML description, in terms of set of Class Diagrams describing objects and relationships; a textual description of the objects and their attributes will be given as well.

In addition to the model, other outputs can be proposals of enhancements to existing APIs as well as new APIs. As an example, a suggestion of API to manipulate some of the identified Framework objects, if useful, can be done.

5.2.2 Framework management tool

Source: Eurescom

Issue/Motivation:

The Information Model for Framework Functions should be accessible from some sort of management tool. To make this possible one should define the API to configure and access the data model.

Information from off-line Service Level Agreements and information needed for on-line Service Agreements should be entered via this API.

Scenarios:

This clause describes a scenario where the extension will make the work of the network operator easier.

When a third party wants to access a service a Service Level Agreement (SLA) between the third party and the network operator. This contract gives a detailed description of all aspects of the deal, such as the extent of the contract (which services should be accessible and the usage allowed), the responsibilities of the network operator and the third party, and actions to be taken if one of the parties does not keep their part of the deal. Many of the points in the SLA needs to be transferred to the Framework so that it can supervise that the contract is respected by the third party and also take action if the contract is not respected. With a Framework Management Tool API (together with a Framework Management Tool) this process would be easier and the possibility to transfer the data from one Framework to another Framework (i.e. if one wants to change vendor) would be there.

The advantages lie in easier management of Parlay/OSA access:

- Quick and easy set-up of access from new third parties.
- Notification (or service denial) if a third party reaches its allowed use, can be sent to both the network operator and the third party.
- Easy change from one vendor to another, because the Management tool uses specific standardized interfaces.

Requirements description:

The Framework management tool API must be designed to be able to access and make changes to data concerning the agreements between network operator and third party whilst the Framework is running.

The Framework management tool API must be designed to be independent of vendor and implementation language used in the management application and underlying network.

The Framework management tool API should provide access to the Framework Information Model that should contain these (and possibly other) aspects of the Service Level Agreement:

- The SCSs open to access by the third party (or part of SCSs functionality).
- Maximum traffic/usage of SCSs.
- Actions to be taken when maximum traffic/usage is reached.
- Restrictions in the use of methods.

Restrictions in visibility of certain information (parameters) and restrictions in modification of certain parameters.

Possible solution and further consideration:

The definition of the Framework management tool API should be based on the Framework Information Model.

5.2.3 Enhancements on event notification handling

Source: Parlay

Requirements description:

The behaviour of event notification handling shall be described in more detail:

- Define rules on how to deal with multiple applications registered on the same event criteria.
- Define behaviour of the Parlay Gateway when multiple applications invoke methods that may interrupt processing of the same call. Those methods include `enableCallNotification` and `routeReq` enabling interception of a controlled call.
- Define rules on how to deal with events that are matching multiple event criteria e.g. a service registered on a criterion on CLI and another on a criterion on dialled number.

Possible solution and further consideration:

Define a general mechanism, based on policies/rules, to allow a better control of event-based application activation/trigger, and event notification;

Let IpCallControlManager have a service interaction management capability and define behaviour when:

- Requesting event report (e.g. enableCallNotification, routeReq).
- Notifying event (e.g. callEventNotify, routeRes).
- Requesting processing a call (e.g. routeReq, release).

The service interaction management capability could:

- introduce some ordering;
- allow the forwarding to all the registered applications;
- apply some load balancing criteria at application side.

5.2.4 Framework operator administration interfaces

Source: Lucent

Issue/Motivation:

Service Type Management: Currently, the Parlay/OSA/ETSI framework allows for dynamic registration of SCSs that have not been known at deployment time. However, the SCSs have to register for a certain service type, which must be known before a SCSs can register itself for that service type. In other words, new SCSs can be added to a framework without any configuration effort, but only for service types the framework knows about.

Adding this feature means that it allows SCSs to add an appropriate service type before registering themselves and that would improve the concept of dynamic configuration of the framework.

Framework/Service interface expansion:

Service Property: There are a number of ways that Service Properties are currently used (Parlay 3.0) but once a Service is registered the properties cannot be changed. There are scenarios where changing them may be desirable, e.g. when a service is selected based on a certain usage price (specified as a service property), and then the provider decides to offer a discount, there should be a means for the SCS to inform the application about it.

Criteria matching while doing service discovery is primitive in Parlay 3.0. It should be possible for a Client to define how the requested properties should be matched to the registered properties.

Contract/SLA Format: There is currently no definition for the structure of contracts/SLAs. A "default" one may be desirable as it could allow more automated discovery and signing of agreements.

Requirements description:

Service Type Management:
Interfaces that enable management (creation/deletion etc) of Service Types shall be added. Lucent has already proposed a draft API for this.

Framework/Service Interface expansion:
Parlay 3.0 saw the replacement of the Service Factory with the Service Instance Lifecycle Manager. This currently has created and destroy methods. This interface could be extended to provide suspend/resume methods, for example.

Service Properties:

- There shall be a means at the framework-service interface as well as at the framework-application interface to propagate a change of service property values.
- There shall be a means for the application to request how matching of requested service properties against registered service properties is done by the framework.

Contract/SLA Format:
A "default" structure for SLAs shall be defined.

5.3 Call Control

5.3.1 IM Session control functions

Source: SA1 [2]

Issue/Motivation:

The OSA APIs shall provide a means to control sessions in the IP multi-media domain (IMS).

Requirements description:

- **Session control**
  - **Create multi-media sessions**
    The application shall be able to establish sessions between two or more parties with certain media capabilities. The application may add and remove parties at any time for any ongoing session. An application may add additional sessions with certain media capabilities between any parties already involved in an ongoing session. Sessions with multiple parties may lead to the creation of a Multi-media Conference Call. This can either be an ad-hoc conference creation or it can refer to resources that were reserved in advance.
  - **Release multi-media calls**
    This provides the ability for an application to force the release of a multi-media session. This may be limited to the release of certain parties from the session or may be the release of all the parties.
  - **Party join/leave control**
    The application shall be capable to be informed when a new call party wants to join/leave the conference. It shall be possible for the application to allow or reject the inclusion of the new party to a conference.
- **Media control**
  - **Control media channels**
    The application shall have the ability to control media channels originated by (or on behalf of) a user or media channels terminated to a user. This control includes, but is not limited to the barring of a media channel request, allowing the media channel establishment to continue with or without modified information, addition or removal of additional media channels, temporarily suspend a media channel (place on hold), open, close or modify the parameters of the media channels.
  - **Relinquish control over specific media channels**
    This allows an application to relinquish control over the media stream. When it relinquishes control over certain media channels it does not lose control over the entire session.

- **Reserve/Free conference resources**
  The application shall be able to reserve resources in the network or free earlier reserved resources for a conference in advance.

- **Information**

- **Request notification of media channel events**
  The application shall be able to request notification of certain events associated with a type of media channel. Events include, but not limited to: a user initiating or closing a session, an incoming session request to user or a terminating user unable to accept an incoming session request.

- **Monitoring of media channels**
  The application shall be able to request all the media channels currently available on a call. In addition the application must be able to monitor on the opening and closing of channels for media for a specified call.

5.3.2 Packet switching Call Control functions

Source: SA1 [2]

Issue/Motivation:

This clause details with packet switched call control functions. The purpose of this function is to allow applications to control and monitor GPRS sessions. A GPRS Session may consist of one or more GPRS PDP context.

Requirements description:

Applications should have the ability to:

- *Release a PDP context:* This provides the ability for the application to force a PDP context to be released. The application may provide an indication of the reason for release of the PDP context.
- *Control a PDP context:* This provides the ability for an application to modify the information pertaining to the PDP context at the time of establishment. The application may also allow the PDP context to continue with or without the modified information pertaining to the PDP context. The application shall have the ability to request events to be observed by the network and reported back to the application.
- *Monitor a PDP context:* This provides the ability for an application to monitor for PDP context duration and tariff switching moments.. An application may specify a threshold for the duration of a PDP context or a part thereof. The application shall have the ability to grant new thresholds when the expiry of a previously set threshold has been reported to the application.
- *Monitor a GPRS session:* This provides the ability for an application to monitor for GPRS session data volume. An application may specify a threshold for the amount of data allowed to be transferred within a GPRS session. The application shall have the ability to grant new thresholds when the expiry of a previously set threshold has been reported to the application.

5.4 Terminal capabilities

5.4.1 Discovery of client terminal capabilities

Issue/Motivation:

This provides the ability for an application to discover terminal/device capabilities and enable it to make intelligent decisions.

Requirements description:

OSA shall enable an application to obtain information about a terminal's capabilities. Terminal capabilities include: terminal hardware, terminal software, and terminal browser.

5.5 User interaction

5.5.1 Interact with a user

Issue/Motivation:

This provides the ability for an application to interact with a user. An application may be able to send specific information to the user using any media of its choice (text, video, redirection to a web page etc.) and may request the collection of data from the user in a specific media format (voice, DTMF etc.).

5.6 Charging issues related to Call Control

Source: Siemens

5.6.1 GCC

A question that needs to be considered here is should we leave the existing Call Control interfaces as is and ensure that any new call control capabilities are placed in new object classes. These new classes should then be subclasses of, for example, MPCC.

- There is a need to consider extensions to the charging mechanisms in respect of areas such as Media, user Interaction and any other objects that may be identified.

- It should be considered whether or not charging and supervision functionality should be separated from the Call Control API and made more generic. This would allow it to be more applicable to other services such as user interaction and data session. Till now a charging SCF has been defined handling Amount and Unit charging. It is therefore proposed to add Usage sessions and to inherit these sessions at the application and service level so that the charging and supervision functionality can be removed from the Call Control APIs and the other APIs.

- It is proposed to add methods, to request and report charging information. During a call several communication configurations can be established. For each communication configuration it may be possible to request and report charging information allowing the methods to be invoked on multiple occasions. Presently it is not possible for an application to take into account a charging influence communicated from the destination.

- It is proposed that additions on tariff and subtariff are introduced to allow flexibility and that the parameters for charging are lined up with the content based charging. In order to support multi-media sessions, the possibility to indicate charge per time unit seems to be rather limited. It is also preferable to indicate charging per consumed unit.

It is proposed to add a method addOnCharge() allowing to add on charges on top of the running charges for the usage charging.

- As the Call Control API is independent of the underlying network, it may be appropriate to include a parameter that indicates the Network Operator, since several network operators may be involved in sending charging related information.

5.6.2 Multi media call control

- There should be a way to indicate the QOS for any media, so that end users are provided with the correct grade of service.

- Allow an application to send a multi media message to a subscribed set of users.

5.7 Event notification

Event notification function [1]:

The Event Notification Function shall allow an application to specify the initial point of contact, which it is interested in. The Event Notification Function provides the necessary mechanisms, which enables an application to request the notification of subscriber or network-related event(s). An application may in addition request the cancellation of subscriber or network related event notification. For all subscriber-related events the application shall always specify the subscriber for which the Event Notification Function is valid. Once an application has enabled the notification of event(s), the Event Notification Function shall report the event(s) until such time the application explicitly requests the termination of the event(s) notification.

When the event occurs, the application that requested the event is informed. The notification of the event shall be accompanied by unambiguous information identifying the original request and event related data. For example, in case of an application is interested in "message" the notification to the application shall indicate whether it is incoming or outgoing, in case of chargeable events, the application shall receive details as used at the network to create a Call Detail Record. In this case, processing in the network is not suspended after notification of the event to the application.

The Event Notification Function includes the availability of offering additional criteria to be specified by the application. The set of criteria is individual and may vary for the event requested.

Subscriber related events:

- A chargeable event happens.

  When a chargeable event occurs for a given user and this event is armed by an application, that application shall be notified.

- The Terminal Capabilities are changed.

  When the capabilities of a terminal change (e.g. when a keyboard is attached) and this event is armed by an application, that application shall be notified.

- A change in the presence related information.

  If any presence related information changes (such as one or more presence information attributes or a user's availability), and this event is armed by the application, that application shall be notified. Presence information may be associated with a user, device or service, or may be a more abstract entity that has the ability to report presence information.

NOTE: The ability to support this function is dependent on the ability of a terminal (through e.g. MExE or WAP) to notify changes in its capabilities. Therefore this function will *not* be able to supply event notifications for terminals not supporting notification of their terminal capabilities.

5.8 Network controlled notifications

At present OSA/Parlay applications have to explicitly request for triggers and notifications in the network. This means that the application is responsible for provisioning.

Another option is that the provisioning of triggers is done by the Home Environment and that the application only has to indicate its availability and tell the SCS that it can receive notifications.

For this SA2 has stated the following requirements for OSA Release 5 (see TS 123 127 [2]):

Requirements description:

Specific methods shall be specified in OSA Network Service Capability Features, permitting: (1) An OSA application to request user related event notifications pertaining to any subscribed user for which the service implemented by the application is activated. (2) The OSA SCS to report user related event notifications in which it explicitly identifies the user to which the event applies. (3) An OSA application to request a function to be applied to all current subscribed users for which the service implemented by the application is activated.

5.9 Content based charging

The content based charging reference architecture is given in figure 1 for information.

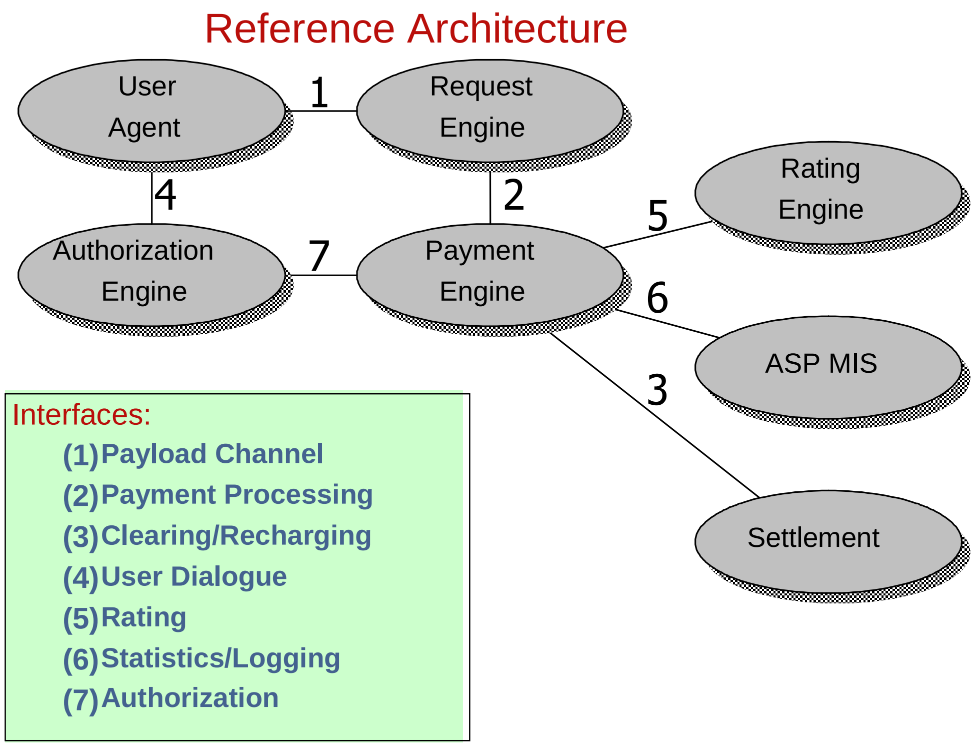

Figure 1

5.9.1 Service properties

Source: Parlay

Issue/Motivation:

In Parlay 3.0, there is the concept of *Service Properties* that provides a standardized format to exchange service configuration data. Service Properties are exchanged in XML format. There are XML elements specific to each Parlay API; these XML elements are defined by an XML-DTD. Currently, there is a Service Property specification for the call control API. They are kept in a single DTD.

Requirements description:

Service Properties should be defined for the Content Based Charging API.

Proposed solution and further considerations:

The particular Service Properties and their exact semantics need to be discussed by the workgroup. The following attributes are given as examples to give an idea what should be configurable by Service Properties:

- Lifetime of a charging session.
- Maximum reservation/maximum charge amount allowed.

5.9.2 User confirmation

Source: Parlay

Issue/Motivation:

It is well understood that in general an implementation of a Payment Engine needs to have a means to request explicit user confirmation from a subscriber before debiting his account. However, with respect to the time and resources available for completing the Content Based Charging API specification for Parlay 3.0, user confirmation has not been investigated so far.

For Parlay 4.0, user confirmation shall be considered in detail. The issues identified so far are discussed below.

Requirements description:

User Confirmation Mechanisms: To better understand the requirements on the Content Based Charging API that are introduced by the need to obtain a user confirmation, appropriate mechanisms that are deployed today will be investigated.

The workgroup should investigate different confirmation mechanisms. A *representative selection of confirmation mechanisms* shall be fixed, similar to the benchmark scenarios that comprise the base of the Parlay 3.0 specification of the Content Based Charging API. Eventually, the existing Content Based Charging API should be enhanced in a backwards-compatible manner.

User Confirmation Support in Content Based Charging API: The confirmation mechanism that suits a given service and user will depend on different factors, one of them being the communication protocol and media utilized between the Request Engine and the User Agent.

It shall be investigated if the existing Content Based Charging API provides sufficient information to the Payment Engine to select and perform an appropriate confirmation mechanisms. If not, appropriate extensions will be introduced for Parlay 4.0.

Proposed solution and further considerations:

The representative selection of confirmation mechanisms could consist of, for instance:

- SMS based confirmation;
- confirmation based on Web/WAP redirect; and
- implicit confirmation based on a user-configured sensitivity level.

For instance, if the Consumer buys physical goods in a shop, a confirmation based on a voice call may be appropriate. If the Consumer is accessing a Web service, a confirmation based on Web redirect will be more convenient for this Consumer. Although the Content Based Charging API should be independent on the confirmation mechanisms available, it still could be useful if the Request Engine has a means to provide information about the communication protocol and media utilized between the Request Engine and the User Agent. The information could be relayed by the Payment Engine towards the Authorization Engine.

The mechanism to be used here could be the Service Parameter argument of all methods. Expected changes will consist of:

- defining new Service Parameter IDs.

5.9.3 Support of roaming/Multi-network scenarios

Source: Parlay

Issue/Motivation:

In Parlay 3.0, no effort has been spent in supporting Content Based Charging across multiple networks. However, in general it is desirable that subscribers who have subscribed as Consumers to one Payment Service Provider can also access services offered in other networks and pay for them through the Content Based Charging Mechanism.

Although the Parlay APIs shall not impose any specific network architecture, the Content Based Charging workgroup shall make sure that the methods on the API map to functionality that is in fact available in the network.

Requirements description:

The workgroup should ensure that Content Based Charging works in Roaming and Multi-Network scenarios.

Proposed solution and further considerations:

This task could possibly comprise the following activities:

- Sketch how a roaming and multi-network scenario could look like. Add this to the benchmark scenarios.
- Check the technical impact on the specification.
- Change the specification if necessary, according to the requirements derived from the benchmark scenario.

5.9.4 Separation of rating and non-rating functionality

Source: Parlay

Issue/Motivation:

When designing the class model for the Content Based Charging API, care has been taken to support a distribution of the Charging Session contexts over multiple servers. This resulted in a separation between a Charging Manager interface (which is typically instantiated once per service), and a Charging Session interface (being instantiated once per service instance).

On the other hand, there are different types of payment operations: Some take a currency amount as input parameter (the rating of the service is done by the service implementation), while some take only events (or units) as input parameters (the rating is then done by the implementation of the Payment Engine). The possibility to mix both types of operations is limited: Once a reservation for a currency amount has been made, no reservation for units is allowed (see the STD in the Parlay 3.0 specification).

However, vendors may wish to host Charging Sessions that do use rating on different systems than the ones that do not use rating. This is currently not possible since upon creation of the Charging Session it is unknown if it will use rating or not.

Requirements description:

The workgroup should discuss if the client should make the intended use of the rating functionality explicit, and if so, provide appropriate means in the specification.

Proposed solution and further considerations:

The desired behaviour could be achieved by adding specific creation methods to the Charging Manager interface.

5.9.5 Split charging

Source: Parlay

Issue/Motivation:

The current specification (Parlay 3.0) of the Content Based Charging API assumes that a single instance of the merchant application (acting as a request engine) serves a single consumer. This is appropriate for many merchant applications, such as video on demand or stock exchange rate information. However, there are cases where a single instance of the merchant application may serve more than a one service user. Examples are multi-user games or conferences. Typically, the costs for the resources consumed by the single service instance will be split amount all service users.

On the other hand, a merchant may show advertisements within its application, and in turn the company that is advertised may subside a certain percentage of the application cost. A consumer connecting to the merchant application pays only part of the costs, while the remainder is paid by the advertised company.

Requirements description:

The Content Based Charging API shall allow clients to specify multiple users within a single charging operation. The charge shall be split among the specified users. The split of the charge between the given user identities could be determined by the request engine or by the payment engine. The given users may either use the client application simultaneously, or may have agreed to subsidies the costs without actually using the application.

Proposed solution and further considerations:

The scenarios described above require that the request engine is able to specify multiple user identities within a single charge request. The split of the charge between the given user identities could be determined by the request engine or by the payment engine.

6 New interfaces and areas of involvement

6.1 Information transfer function

Issue/Motivation:

The Information Transfer function shall enable an application to indicate to a user respectively an application in the UE or USIM about the presence of existing information for her. Physically, this indication may be sent by the underlying network e.g. as a SMS, Instant Messaging, USSD message to the terminal. The Information Transfer function provides the means to inform the underlying network that an indication shall be sent to the user.

NOTE: For 3G release 99 mechanisms like USSD or SMS may be employed to transfer the indication to the user's terminal. For 3G Release 5 IMS Instant Messaging may be employed to transfer indication to the user's terminal.

Requirements description:

The following functions shall be supported:

- *send information notification* provides the means to inform the underlying network that an indication shall be sent to a user respectively an application in the UE or USIM about the presence of existing information for her;
- *request message receipt notification:* the application can request to receive a notification every time a message is received in the mailbox for the user. This allows the application to take the appropriate action, e.g. informing the user.

6.2 Presence related capability functions

The OSA interface shall allow an application access to presence capabilities within the network. Presence related information may be requested or supplied by an OSA application and may include, but not limited to presence information pertaining to the presence service as described in [2] or user availability.

An OSA application may act as a requester of presence information (i.e. act as a watcher) and/or act as a supplier of presence information (i.e. act as a presentity). All the capabilities offered to presence service watchers and presentities are described in [2] and may be offered to OSA applications. In addition to the authorization performed by the OSA Framework, the presence service checks that the application is permitted to access the presence service.

An OSA application may manage or query availability status and/or preferences of a user which may be associated with one or more services (e.g. voice call, IMS sessions, MMS, etc.). Such availability may be determined from a range of existing capabilities.

The following OSA capabilities shall be supported for an application:

- Register as a presentity and/or watcher:

The application shall be able to request the registration as a presentity and/or as a watcher in the presence service. This registration shall include the ability to establish as well as cancel a registration.

- Supply presence related information to the network:

  The application shall be able to supply and/or update presence related information (presence information or availability) at any time. An application may modify the availability of a user.

- Request the querying and/or modification of presence related data:

  The application shall be able to request the querying and/or modification of data other than - presence information related to watchers and/or presentities. Such data includes but is not limited to any access rules pertaining to the presentity to be modified. An application may be able to request the management of availability preferences of a user. Management includes the setting, modification and deletion of availability preferences.

- Request presence related information:

  The application shall be able to request presence related information . The application shall be able to request presence information about a presentity or may request the availability of a user. Such requests may be for the current information, on a periodic basis or for future changes in the presence related information (e.g. arming of event notifications).

- Retrieve watcher information:

  The application shall be able to request watcher information about a presentity.

6.3 Policy management

Applications should have the ability to interact with policy-enabled Service Capability Features in a secure manner. The network policies always take precedence over the application-defined policies.

The interface shall provide sufficient capabilities to enable applications to request:

- To manage the application's policy-related information:

  This allows applications to create, modify and delete policies, policy events and to activate and deactivate policy rules.

- To manage policy event notification:

  This allows applications to register for specific policy events. Once registered for such events, the application shall receive notification of the events until it explicitly requests the termination of the notification request.

- To collect policy statistics:

  This allows an application to collect policy related statistics from the network. Examples include success or failure of operations on policies and time stamps of policy events. This one is not in the existing Parlay doc; the others above are!

6.4 Parlay and SIP

Source: BT Exact

As part of the ongoing work it is imperative that we ensure that the API is able to be mapped to the underlying network technologies.

Requirements description:

Ensure that the API is compatible with SIP: All of the capabilities associated with the API should be able to be transported by SIP where necessary. It may also be appropriate to make changes to the API in line with SIP requirements.

6.5 Inclusion of SOAP/XML as an alternative transport mechanism

Release 3.0 of the Parlay APIs have not effectively embraced the IT and developer communities from a membership and technical content perspective. A UML to XML mapping rules for Parlay 3.0 would revive interest from the IT industries.

Having an XML interface specification would greatly dispel those who say that Parlay is not IT or Internet friendly because it does not have an XML interface specification. We need to consider whether this "mapping" document is provided as part of Parlay 3.1 or created in parallel in Parlay 4.0.

The goal here is to address the IT and developer communities and get them actively participating in the Parlay Group, to work with those industries to make Parlay easier to use, to make Parlay visible, and to dispel myths created by competitive marketing.

In order to achieve a XML support for the current OSA/Parlay APIs some outstanding issues need to be resolved. For some of them a possible solution is already proposed, but nevertheless we feel it is useful to outline the issues and sketch multiple potential solutions.

- Distribution technology for the XML APIs:

  In principle one could define an XML based set of messages, transferred using HTTP, as currently LIF is doing.

- However, SOAP seems to be a more appropriate technology as this is especially designed as XML based request/response protocol. SOAP is not yet an open solution although the W3C Recommendation of SOAP version 1.2 is expected to be available in May 2002.

NOTE: From the SOAP specification: SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

- For the OSA/Parlay API, the most promising option for achieving an XML version of the API seems to be Web Services Description Language (WSDL) as this specifically designed to describe the interface and functionality of a so-called Web-Service. A Web-Service is just a capability that can be invoked via standard internet protocols (e.g. XML/SOAP or XML/HTTP) an as such an implementation of the OSA/Parlay APIs could as well serve as Web-Service. WSDL can be seen as the IDL of web-services and also offers bindings to different distribution protocols like SOAP or HTTP and thus is independent of the actual underlying protocol. Asynchronous call-backs: at the moment there is no defined solution when using SOAP to cope with asynchronous call-backs. A proposal that is one to one in line with the current OSA/Parlay API, where we have object references as parameter of the request message, is to define in the XML schemas an object reference as URL + String. In the request message the object reference would then be provided as a parameter together with all other parameters and in the response message the object reference would be located in the SOAP header.

- Production process of XML APIs: XML version of the API could be produced from the UML model as we are currently producing IDL from the model. An alternative could be that the XML is generated from the IDL as the OMG defines a set of rules to translate IDL into XML format (See OMG ORBOS CORBA/SOAP RFP: Initial RFP: **orbos/00-09-07** and Joint Initial CORBA/SOAP mapping submission: **orbos/01-06-07**), see figure 2. Furthermore, there are already tools available that are able to transform IDL to WSDL.

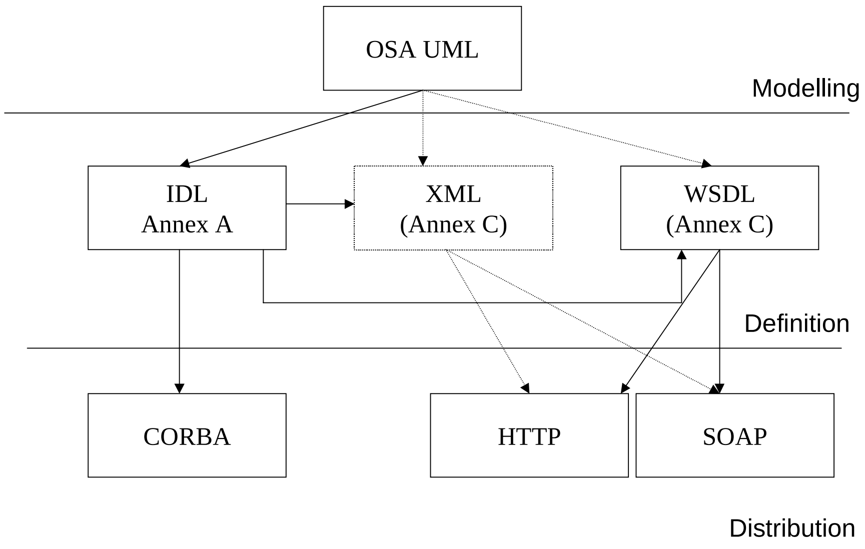

Figure 2

Requirements

- Identify the most suitable solution for incorporating XML support for the current OSA/Parlay APIs. This could be achieved by the joint API group specifying the APIs in WSDL. Or targeting XML/SOAP or XML/HTTP.
- Identify the way to generate WDSL/XML, either via IDL generated from the UML model or directly from the UML model.
- Identify the best solution for providing call back functionality (object references) where we should look for the solution that is optimal alignment with the current APIs.

Annex A (informative): Bibliography

Parlay Specification (Version 2.1): "Framework Interfaces, Client Application View".

History

Document history		
V1.1.1	February 2003	Membership Approval Procedure MV 20030418: 2003-02-18 to 2003-04-18
V1.1.1	April 2003	Publication