3GPP TSG CN Plenary Meeting #25 **NP-040360**
08-10 September 2004, Palm Springs, CA, USA

Source: CN5 (OSA)

Title: Rel-6 Draft TS 29.199-01 to 14 (OSA Parlay X web services) - for Approval

Agenda item: 9.7 (OSA Enhancements [OSA3])

Document for: DECISION

Presentation of Technical Specification to TSG CN

Abstract of document:

- Work done against the WID in NP-040144 (Work Item ID: OSA3).

- This draft TS-family specifies an initial set of ***Parlay X web services***.
  This TS-family also specifies the message(s) exchanged during invocations of each Web Service, by defining the semantics in English and the syntax using W3C WSDL.

Purpose of This Specification:

- The ***OSA APIs*** are designed to enable creation of telephony applications as well as to "telecom-enable" IT applications. IT developers, who develop and deploy applications outside the traditional telecommunications network space and business model, are viewed as crucial for creating a dramatic whole-market growth in next generation applications, services and networks.
- The ***Parlay X web services*** are intended to stimulate the development of next generation network applications by developers in the IT community who are not necessarily experts in telephony or telecommunications. The selection of Web Services should be driven by commercial utility and not necessarily by technical elegance. The goal is to define a set of powerful yet simple, highly abstracted, imaginative, telecommunications capabilities that developers in the IT community can both quickly comprehend and use to generate new, innovative applications.

Changes since last presentation to TSG-CN

- Draft TS 29.199 (as single document) was presented for the 1st time for Info as NP-030552 at CN#22, 12/2003.
- Draft TS 29.199 (Parts 1 to 9) was presented for the 2nd time for Info as NP-040274 at CN#24, 06/2004.
- Draft TS 29.199 (Parts 1 to 14) is submitted for Approval as NP-040360 to CN#25, 09/2004.
- New parts had been added (Parts 10 to 14) as follows:

TS 29.199 specification	CN#24 2nd time for Info	CN#25 for Appr.	Note
Part 1: Common	ver. 1.0.3	ver. 2.0.0	
Part 2: Third party call	ver. 1.0.3	ver. 2.0.0	
Part 3: Network-initiated third party call	ver. 1.0.3	ver. 2.0.0	
Part 4: Short Message Service (SMS)	ver. 1.0.3	ver. 2.0.0	
Part 5: Multimedia Message Service (MMS)	ver. 1.0.3	ver. 2.0.0	
Part 6: Payment	ver. 1.0.3	ver. 2.0.0	
Part 7: Account management	ver. 1.0.3	ver. 2.0.0	
Part 8: User status	ver. 1.0.3	ver. 2.0.0	
Part 9: Terminal location	ver. 1.0.3	ver. 2.0.0	
Part 10: Call handling	Not available	ver. 1.0.0	new
Part 11: Audio call	Not available	ver. 1.0.0	new
Part 12: Multimedia Conference	Not available	ver. 1.0.0	new
Part 13: Address List Management	Not available	ver. 1.0.0	new
Part 14: Presence	Not available	ver. 1.0.0	new

Outstanding Issues:

None.

Contentious Issues:

None.

3GPP TS 29.199-1 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 1: Common (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 7
3.1 Definitions 7
3.2 Abbreviations 7
4 Use of Web Services Technologies 7
4.1 Web Service Message Content 7
4.1.1 SOAP 7
4.1.2 XML 7
4.1.3 HTTP 7
4.2 Web Service Interface Definitions 7
4.2.1 WSDL 8
4.3 Security for Parlay X Web Services 8
4.4 XML Data Types 8
5 Detailed Service Description 8
5.1 Address Data Items 8
5.2 Charging 8
5.2.1 Charging Data Type 9
5.3 Exception Definition 9
5.4 Service Exception 10
5.5 Policy Exception 10
6 Namespaces 10
7 Sequence Diagrams 10
8 XML Schema Data Type Definition 10
8.1 TimeMetrics enumeration 10
8.2 TimeMetric structure 11
8.3 ChargingInformation structure 11
8.4 ServiceError structure 11
8.5 SimpleReference structure 11
9 Web Service Interface Definition 12
10 Fault Definitions 12
10.1 ServiceException 12
10.1.1 SVC0001: Service error 12
10.1.2 SVC0002: Invalid input value 12
10.1.3 SVC0003: Invalid input value with list of valid values 12
10.1.4 SVC0004: No valid addresses 12
10.1.5 SVC0005: Duplicate correlator 13
10.1.6 SVC0006: Invalid group 13
10.1.7 SVC0007: Invalid charging information 13
10.2 PolicyException 13
10.2.1 POL0001: Policy error 13
10.2.2 POL0002: Privacy error 13
10.2.3 POL0003: Too many addresses 14
10.2.4 POL0004: Unlimited notifications not supported 14
10.2.5 POL0005: Too many notifications requested 14
10.2.6 POL0006: Groups not allowed 14
10.2.7 POL0007: Nested groups not allowed 14
10.2.8 POL0008: Charging not supported 14
10.2.9 POL0009: Invalid frequency requested 14
10.3 Fault Number Ranges by Service 15

11 Service Policies 15
12 WSDL Usage and Style 15
12.1 Service Definition and Documents 15
12.1.1 Interface Sets 16
12.1.2 Preparing for Document Definition 16
12.1.3 Documents 16
12.1.3.1 Types Definition Document 16
12.1.3.2 Shared Faults Document 17
12.1.3.3 Service Interface Document 17
12.1.3.4 Service Bindings Document 17
12.1.4 Document Separation Rationale 17
12.1.5 Document Version Identifier 18
12.1.6 Document Naming Example 18
12.1.7 Service Definitions for Notification Patterns 18
12.2 Namespaces 19
12.2.1 Namespaces for Parlay X Web Services 19
12.2.2 Use of Namespaces 19
12.2.3 Namespace Elements 19
12.2.4 Common Namespaces 19
12.2.5 Target Namespace 20
12.2.6 WSDL and Schema Namespaces 20
12.2.7 Local Namespace Use 20
12.2.8 Examples 21
12.3 Authoring Style – Document Content and Names 21
12.3.1 General WSDL Document Information 21
12.3.2 Names 21
12.3.3 Case Usage for Names 22
12.3.4 Naming Conventions for Special Names 22
12.3.5 Document Layout 22
12.4 Data Type Definitions 23
12.4.1 Types Section Declaration 23
12.4.1.1 Optional Elements 23
12.4.1.2 Nillable Elements 23
12.4.1.3 User Defined Simple Data Types 23
12.4.1.4 Data Structures 23
12.4.1.5 Enumerations 23
12.4.1.6 Unions 24
12.4.1.7 Web Service References 24
12.5 Messages and Interfaces (PortTypes) 25
12.5.1 Messages 25
12.5.1.1 Document Style Web Services 25
12.5.2 Interfaces (PortTypes) 25
12.5.3 Faults (Exceptions) 25
12.6 Bindings and Service Definitions 26
12.6.1 Binding 26
12.6.2 Service Definition 26

Annex A (normative): WSDL for Common Data Definitions 27

Annex B (informative): Change history 28

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 1 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	**Common**
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 1 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Common aspects of the Parlay X Web Services.The following are defined here:

- Name spaces
- Data definitions
- Fault defenitions
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] RFC 2806 - URLs for Telephone Calls, available at www.ietf.org/rfc/rfc2806.txt

[7] RFC 3261 - SIP: Session Initiation Protocol, available at www.ietf.org/rfc/rfc3261.txt

[8] WS-I Basic Profile 1.0, available at www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[9] Web Services Description Language 1.1, available at www.w3.org/TR/2001/NOTE-wsdl-20010315

[10] WS-Security, available at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[11] XML Digital Signature, available at http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

3 Definitions and Abbreviations

3.1 Definitions

Application: a computer program that accesses a Web Service.
Web Service: A software system designed to support interoperable machine-to-machine interaction over a network.
Web Service Provider: An entity which provides Web Services interfaces to capabilities offered.
Web Service Requester: An entity which operates Applications that access Web Services.

3.2 Abbreviations

3GPP	Third Generation Partnership Project
ETSI	European Telecommunications Standards Institute
IP	Internet Protocol
IT	Information Technology
OASIS	Organization for the Advancement of Structured Information Standards
OSA	Open Service Access
PSTN	Public Switched Telephone Network
RFC	Request for Comment
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SOAP	Not an acronym, protocol used for XML messaging
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WSDL	Web Service Definition Language
WS-I	Web Services-Interoperability Organization
XML	Extensible Markup Language

4 Use of Web Services Technologies

4.1 Web Service Message Content

4.1.1 SOAP

All Web Service messages SHALL send and accept messages that conform to the SOAP use defined in the WS-I Basic Profile [8], using the document/literal encoding style.

4.1.2 XML

All Web Service messages SHALL send and accept messages that conform to the XML use defined in the WS-I Basic Profile [8].

4.1.3 HTTP

All Web Service messages SHALL send and accept messages that conform to the HTTP use defined in the WS-I Basic Profile [8].

4.2 Web Service Interface Definitions

All Parlay X Web Services are defined according to the following.

4.2.1 WSDL

All Web Service interfaces SHALL be defined using WSDL 1.1 as defined in the WSDL specification [9] and be conformant to the WSDL use defined in WS-I Basic Profile [8].

See Section 15 for detailed information on the WSDL style to be followed by Parlay X Web Services.

4.3 Security for Parlay X Web Services

If a message contains an identifier and/or credentials representing the sender of the message then these SHALL be provided in a manner prescribed by WS-Security[10].

Encryption of message content MAY be required by the Web Service Provider. If this is required, then this SHALL be accomplished in one of the following manners,

- Use of a Virtual Private Network, to be administered independent of the Web Service implementation
- Use of Transport Level Security using HTTP over TLS as specified in the WS-I Basic Profile [8]

Integrity of the message content MAY be required by the Web Service Provider. If this is required, then this SHALL be accomplished using XML Digital Signature [11].

4.4 XML Data Types

Where possible standard XML Schema data types are used, as defined in section 3 (Built-in datatypes) in XML Schema [5].

5 Detailed Service Description

5.1 Address Data Items

Addresses, unless the specification provides specific additional instruction, MUST conform to the address portion of the URI definition provided in [RFC 2806] for ‘tel:’ addresses or [RFC 3261] for ‘sip:’ addresses. Optional additions to the address portion of these URI definitions MUST NOT be considered part of the address accepted by the Parlay X Web Services interfaces, and an implementation MAY choose to reject an address as invalid if it contains any content other than the address portion.

When processing a ‘tel:’ URI, as specified in [RFC 2806], Parlay X Web Services MUST accept national addresses (those not starting with ‘+’ and a country code) and MUST accept international addresses (those starting with ‘+’ and a country code).

When specified in the definition of a service operation, the URI may contain wildcard characters in accordance with the appropriate specification (i.e. RFC 2806 or RFC 3261).

5.2 Charging

Web Services may use a Web Service Provider to deliver content or function. In some cases, the producer of the content or capability will wish to use a bill-on-behalf-of capability offered by the Web Service Provider to charge for the content/function provided. For those services where the charge is part of a single activity, providing the charging related information as part of the message is very efficient.

An example is a messaging service, where a sports business collects information and distributes short messages with sports scores to its subscribers. The sports business has an agreement with a Web Service Provider where the charges for the messages are included in the bill provided by the Web Service Provider (thus the Web Service Provider is billing on behalf of the sports business.

To enable this capability to be provided across a variety of services in a consistent manner, thus making implementation easy and efficient, the information to be provided in the Web Service message for charging information is defined as part of the Parlay X Web Services Framework.

5.2.1 Charging Data Type

The charging information is provided in an XML data type, using the following schema.

```
<xsd:complexType name="ChargingInformation">
   <xsd:sequence>
      <xsd:element name="description" type="xsd:string"/>
      <xsd:element name="currency" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
      <xsd:element name="amount" type="xsd:decimal" minOccurs="0"
maxOccurs="1"/>
      <xsd:element name="code" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

The application accessing the Web Service provides this information,

- Description text, which will often be used to provide billing text. This text does not have specific required content, but would likely include information on the business, the content or service provided, and a transaction identifier. Credit card statements are a good example of description text provided by different companies.
- Currency in which the charge is to be applied. Values for the currency field are defined by ISO 4217.
- Defines the amount to be charged.
- Code specifies a charging code which references a contract under which this charge is applied. The code identifier is provided by the Web Service Provider.

The charging information provided may not be acceptable to the Web Service Provider. For example, the Web Service Provider may limit the amount that may be specified for a particular Web Service or for a particular Web Service Requester. If the information provided is not acceptable, an appropriate fault message may be returned to the Web Service Requester (SVC0007 is defined as a generic charging fault).

5.3 Exception Definition

Exceptions are defined with three data items.

The first data item is a unique identifier for the message. This allows the receiver of the message to recognize the message easily in a language-neutral manner. Thus applications and people seeing the message do not have to understand the message text to be able to identify the message. This is very useful for customer support as well, since it does not depend on the reader to be able to read the language of the message.

The second data item is the message text, including placeholders (marked with %) for additional information. This form is consistent with the form for internationalization of messages used by many technologies (operating systems, programming environments, etc). Use of this form enables translation of messages to different languages independent of program changes. This is well suited for Web Services messages, as a programming language is not defined.

The third data item is a list of zero or more strings that represent the content to put in each placeholder defined in the message in the second data item.

5.4 Service Exception

When a service is not able to process a request, and retrying the request with the same information will also result in a failure, and the issue is not related to a service policy issue, then the service will issue a fault using the ServiceException fault message. A Service Exception uses the letters 'SVC' at the beginning of the message identifier.

Examples of service exceptions include invalid input, lack of availability of a required resource or a processing error.

5.5 Policy Exception

When a service is not able to complete because the request fails to meet a policy criteria, then the service will issue a fault using the PolicyException fault message. To clarify how a Policy Exception differs from a Service Exception, consider that all the input to an operation may be valid as meeting the required input for the operation (thus no Service Exception), but using that input in the execution of the service may result in conditions that require the service not to complete. A Policy Exception uses the letters 'POL' at the beginning of the message identifier.

Examples of policy exceptions include privacy violations, requests not permitted under a governing service agreement or input content not acceptable to the service provider.

6 Namespaces

The namespace for the common data types is,

http://www.csapi.org/schema/common/v2_0

The namespace for the common faults is,

http://www.csapi.org/wsdl/common_faults/v2_0

The 'xsd' namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name 'xsd' is not semantically significant.

7 Sequence Diagrams

Not applicable.

8 XML Schema Data Type Definition

8.1 TimeMetrics enumeration

List of time metric values.

Enumeration	Description
Millisecond	Millisecond
Second	Second
Minute	Minute
Hour	Hour
Day	Day
Week	Week

Month	Month
Year	Year

8.2 TimeMetric structure

For services that provide service based on a time interval or duration or similar metric, this type is used to specify the time metric.

Element Name	Element Type	Description
Metric	TimeMetrics	Metric to use for time measurement
Units	xsd:int	Number of units of TimeMetrics

8.3 ChargingInformation structure

For services that include charging as an inline message part, the charging information is provided in this data structure.

Element Name	Element Type	Description
Description	xsd:string	Description text to be use for information and billing text
Currency	xsd:string	Currency identifier as defined in ISO 4217 (optional)
Amount	xsd:decimal	Amount to be charged (optional)
Code	xsd:string	Charging code, referencing a contract under which the charge is applied (optional)

8.4 ServiceError structure

Some services that process requests for both single addresses and group of addresses return a fault message for the single request, and a data item for the group response. This data structure allows the data item returned for a group response to contain the same level of information as the fault message response.

Element Name	Element Type	Description
MessageId	xsd:string	Message identifier (take from fault definitions)
Text	xsd:string	Message text, with replacement variables marked with %#
Variables	xsd:string [0..unbounded]	Variables to substitute into Text string

8.5 SimpleReference structure

For those services that require a reference to a Web Service, the information required to create the endpoint information is contained in this type.

Element Name	Element Type	Description
Endpoint	xsd:anyURI	Endpoint address
InterfaceName	xsd:string	Name of interface
Correlator	xsd:string	Correlation information

9 Web Service Interface Definition

Not applicable.

10 Fault Definitions

10.1 ServiceException

Faults related to the operation of the service, not including policy related faults, result in the return of a ServiceException message. Service exception messages use the reserved message identifier 'SVC', and are defined with numbers from 0001 to 0999, with numbers 0001 to 0199 reserved for common exceptions and 0200 to 0999 for Parlay X Web Services specification use. Numbers from '1000' to '9999' may be used by third parties.

Element Name	Element Type	Description
MessageId	xsd:string	Message identifier, with prefix SVC
Text	xsd:string	Message text, with replacement variables marked with %#
Variables	xsd:string [0..unbounded]	Variables to substitute into Text string

10.1.1 SVC0001: Service error

MessageId	SVC0001
Text	A service error occurred. Error code is %1.
Variables	%1 Error code from service - meaningful to support, and may be documented in product documentation.

10.1.2 SVC0002: Invalid input value

MessageId	SVC0002
Text	Invalid input value for message part %1.
Variables	%1 –message part.

10.1.3 SVC0003: Invalid input value with list of valid values

MessageId	SVC0003
Text	Invalid input value for message part %1, valid values are %2.
Variables	%1 –message part %2 – list of valid values

10.1.4 SVC0004: No valid addresses

MessageId	SVC0004
Text	No valid addresses provided in message part %1

Variables	%1 – message part

10.1.5 SVC0005: Duplicate correlator

MessageId	SVC0005
Text	Correlator %1 specified in message part %2 is a duplicate.
Variables	%1 – correlator %2 – message part

10.1.6 SVC0006: Invalid group

MessageId	SVC0006
Text	Group %1 in message part %2 is not a valid group.
Variables	%1 – identifier for the invalid group %2 – message part

10.1.7 SVC0007: Invalid charging information

MessageId	SVC0007
Text	Invalid charging information.
Variables	None

10.2 PolicyException

Faults related to policies associated with the service, result in the return of a PolicyException message. Policy exception messages use the reserved message identifier 'POL', and are defined with numbers from 0001 to 0999, with numbers 0001 to 0199 reserved for common exceptions and 0200 to 0999 for Parlay X Web Services specification use. Numbers from '1000' to '9999' may be used by third parties.

Element Name	Element Type	Description
MessageId	xsd:string	Message identifier, with prefix POL
Text	xsd:string	Message text, with replacement variables marked with %#
Variables	xsd:string [0..unbounded]	Variables to substitute into Text string

10.2.1 POL0001: Policy error

MessageId	POL0001
Text	A policy error occurred. Error code is %1.
Variables	%1 Error code from service - meaningful to support, and may be documented in product documentation.

10.2.2 POL0002: Privacy error

MessageId	POL0002

Text	Privacy verification failed for address %1, request is refused.
Variables	%1 – address privacy verification failed for.

10.2.3 POL0003: Too many addresses

MessageId	POL0003
Text	Too many addresses specified in message part %1.
Variables	%1 – message part

10.2.4 POL0004: Unlimited notifications not supported

MessageId	POL0004
Text	Unlimited notification request not supported.
Variables	None.

10.2.5 POL0005: Too many notifications requested

MessageId	POL0005
Text	Too many notifications requested.
Variables	None.

10.2.6 POL0006: Groups not allowed

MessageId	POL0006
Text	Group specified in message part %1 not allowed.
Variables	%1 – message part

10.2.7 POL0007: Nested groups not allowed

MessageId	POL0007
Text	Nested group specified in message part %1 not allowed.
Variables	%1 – message part.

10.2.8 POL0008: Charging not supported

MessageId	POL0008
Text	Charging is not supported.
Variables	None.

10.2.9 POL0009: Invalid frequency requested

MessageId	POL0009
Text	Invalid frequency requested.
Variables	None.

10.3 Fault Number Ranges by Service

The following table includes fault number ranges are reserved for use by specific Parlay X Web Services.

Web Service	SVC range	POL range
Third Party Call	0260-0264	
Multimedia Conference		0240-0244
Short Messaging	0280-0284	
Terminal Status		0200-0204
Terminal Location	0200-0204	0230-0234
Payment	0270-0274	
Account Management	0250-0254	0220-0224
Address List Management		0210-0214
Presence	0220-0224	

11 Service Policies

Not applicable.

12 WSDL Usage and Style

Parlay X Web Services definitions,

- SHALL specify services using document forms as described in 15.1
- SHALL use namespaces as defined in 15.2
- SHALL follow the authoring style as defined in 15.3
- SHALL follow data type definitions as defined in 15.4
- SHALL define messages and interfaces as defined in 15.5 using document/literal definitions
- SHALL define bindings and services as defined in 15.6 using document/literal defintions

12.1 Service Definition and Documents

Service definitions are expressed using the facilities of WSDL. While it is possible to produce a single document that represents an entire service definition, this is not a desirable approach for any non-trivial Web Service.

Decomposition provides the following benefits:

- XML Schema is used for data type definitions
- Faults that are shared across interfaces are defined independently
- Service interface definitions are defined independent of bindings
- Bindings are defined independently and consistent with UDDI best practices

Following these conventions improves the overall definition and maintenance process and improves deployment by supporting separation of interface and binding.

12.1.1 Interface Sets

A Web Service definition may contain one or more interfaces (or portTypes in WSDL 1.1). The characteristics of the Web Service being considered will determine whether one interface or multiple interfaces are appropriate.

The term *Interface Set* will be used to describe the group of interfaces that comprise a Web Service. The *Interface Set* provides a mechanism to group a set of related interfaces using well defined conventions for document and namespace naming.

For reference, other technologies group related interfaces using 'module' or 'package' conventions, achieving a similar result for organizing related interfaces.

12.1.2 Preparing for Document Definition

To provide a consistent use of naming within document sets, and across document sets, a number of conventions are defined that rely on a small amount of preparation to be done before creating the documents.

For each *Interface Set*, a *Base Name* is selected. For each interface within the *Interface Set*, a *Short Name* is selected. These names will be used as part of a common naming convention for the related set of documents defined and for definition naming within the documents. This approach ensures name consistency through Web Service evolution, whether it starts with one interface or multiple interfaces.

An example will demonstrate the naming convention. A group of interfaces for a short messaging service (SMS) are defined. This Web Service contains multiple interfaces.

- An *Interface Set* is defined (SMS Interface Set)
- The *Base Name* for the *Interface Set* is assigned the name 'sms'
- Each interface within the *Interface Set* is assigned a *Short Name*
    - The SendSms interface is assigned the *Short Name* 'send'
    - The RetrieveSms interface is assigned the *Short Name* 'retrieve'

Base Names and *Short Names* are always defined using only lower case letters, numbers or underscore characters. They must not start with a non-alphabetic character. An underscore should be used to separate words when the name consists of multiple words. These restrictions apply since these names are used in the construction of file names and URI content.

With these preparations complete, the document set may be created.

12.1.3 Documents

There are four document types that can be utilized in a Web Service definition. Each has a specific role, and contributes to the goal of supporting a well organized and useful decomposition of the individual elements of a Web Service definition.

12.1.3.1 Types Definition Document

The *Type Definitions Document* contains data type definitions within a schema namespace.

When the document is related to a specific *Interface Set*, it will use the *Base Name* with the suffix '_types' and the extension '.xsd'. When the *Type Definitions Document* is used across multiple *Interface Sets*, it will use an independent name with the suffix '_types' and the extension '.xsd'.

This document is optional, since not all services will define new data types.

12.1.3.2 Shared Faults Document

The *Shared Faults Document* contains fault definitions that are shared across multiple interfaces in an *Interface Set*, or across *Interface Sets*.

The faults are defined within their own namespace within the WSDL definition namespace. The document name for this document will use the suffix '_faults' and the extension 'wsdl'. The first part of the name of the document is based on its usage, with the following guidance;

- If it is used by multiple *Interface Sets*, an independent name reflective of the faults defined will be chosen by the author
- If it used only by multiple interfaces within an *Interface Set*, then the *Base Name* will be used for the first part of the name.

This document is optional, since not all WSDL definitions will define faults that are shared with other WSDL definitions.

12.1.3.3 Service Interface Document

The *Service Interface Document* contains the message and interface (portTypes in WSDL 1.1) definitions. One interface definition is included in each document. This document may import *Type Definition Documents* and *Shared Faults Documents*. This document may be used for a variety of *Service Bindings Documents* without change.

The document name for this document will use the suffix '_interface' and the extension 'wsdl'. The name of the document is determined as follows;

- For each interface in an *Interface Set*, the name is a combination of the *Base Name* followed by an underscore followed by the *Short Name* for the interface defined in this document. Thus multiple documents will have the same *Base Name* as the first portion of the name and the individual interface *Short Name* as the second portion

12.1.3.4 Service Bindings Document

The *Service Bindings Document* contains both the binding to be used and the service definition associated with the binding. One service definition is defined in each document. This document imports one *Service Interface Document*.

The document name for this document will use the suffix '_service' with the extension 'wsdl'. Optionally, text representing the specific binding may be added immediately before the '_service' suffix. The name of the document is determined as follows;

- For each interface in an *Interface Set*, the name is a combination of the *Base Name* followed by an underscore followed by the *Short Name* for the interface defined in this document. Thus multiple documents will have the same *Base Name* as the first portion of the name and the individual interface *Short Name* as the second portion

12.1.4 Document Separation Rationale

The four document types approach satisfies a number of desirable goals for WSDL creation, use and maintenance.

- Types and shared faults are defined in common documents, eliminating redundant definitions
- Interfaces are defined in individual documents, providing easier reading (only relevant message definitions in same document), while using a naming convention that group related interfaces together
- Services are defined in individual documents, providing easy consumption by service registries and easy creation of alternate binding documents. Like the interface documents, the naming conventions for these documents group related services together.

By following this approach, the document decomposition supports modularization for reuse goals, in a manner that reflects a useful level of granularity, and useful document form for use with tools and for deployment use.

12.1.5 Document Version Identifier

Just like namespaces may have naming conflicts, document names may also have naming conflicts. It is not always predictable how documents will be stored and used, or when multiple versions of a service may be co-deployed. For this reason, documents may include version identifiers in their naming.

Documents may be assigned a version identifier, corresponding to version information provided in the namespace (see next chapter for more information on the version identifier).

If used, the identifier is added to the end of the name following an underscore. For example, a namespace version of v2_0 would be expressed as _2_0 added to the end of the document name and before its extension.

12.1.6 Document Naming Example

Using the SMS Interface Set described previously, the following document set would be produced. Additional assumptions for this example are that there are some data type definitions and that multiple interfaces in the *Interface Set* use a common set of faults.

The names provided include the use of version identifiers, where this Web Service is at the v1_0 level.

One *Type Definitions Document* – sms_types_1_0.xsd

One *Shared Faults Document* – sms_faults_1_0.wsdl

Two *Service Interface Documents* – sms_send_interface_1_0.wsdl and sms_retrieve_interface_1_0.wsdl

Two *Service Bindings Documents* – sms_send_service_1_0.wsdl and sms_retrieve_service_1_0.wsdl

The two *Service Interface Documents* import the *Type Definition Document* and *Shared Faults Document*. The two *Service Bindings Documents* import their respective *Service Interface Document*.

12.1.7 Service Definitions for Notification Patterns

A *Service Interface Document* provides the messages and interfaces for a Web Service. It does not distinguish any deployment relationship, though there are specific uses intended for some Web Services definitions.

A common message pattern, defined in the Message Patterns section, is notification. The Web Service has a corresponding facility, such as a Web page, that provides the information required to define the notification, and a WSDL definition that represents the notification definition.

For message patterns that include notifications, the Web Service definition approach is the same, but the roles of the Web Service Provider and Web Service Requestor become a peer-to-peer or a producer-consumer relationship instead of a requestor-provider relationship.

To provide a practical example, the SMS Interface Set described previously will be extended to include two additional interfaces – RegisterSms and SmsNotify. The interfaces will use the *Short Names* ‘register’ and ‘notify’ respectively.

The RegisterSms interface will reuse the current *Type Definitions Document* (sms_types.xsd) and *Shared Faults Document* (sms_faults.wsdl), and adds its own *Service Interface Document* (sms_register_interface.wsdl) and *Service Bindings Document* (sms_register_service.wsdl).

The SmsNotify interface will reuse the current *Type Definitions Document* (sms_types.xsd) and *Shared Faults Document* (sms_faults.wsdl), and adds its own *Service Interface Document* (sms_notify_interface.wsdl) and *Service Bindings Document* (sms_notify_service.wsdl).

The RegisterSms interface is deployed in the same manner as the SendSms and RetrieveSms services at the Web Service Provider. The Web Service Requestor uses the RegisterSms interface to indicate the criteria to be used to determine when a notification is appropriate to send.

Although the notification will be delivered to the entity that had been using the SMS Interface Set in a requestor role, the definition of the SmsNotify service is consistent with the other services in the *Interface Set*. The only difference is that at the time of deployment, the SmsNotify implementation will be deployed in the environment of the entity that deploys the requester side of the other SMS Interface Set interfaces.

Following this approach, a consistent use of document conventions simplifies the process of defining Web Services, regardless of the deployment configurations or roles that deployed services may assume.

12.2 Namespaces

The definitions tag has a number of attributes for namespace definitions. These definitions will include a set of common definitions and WSDL specific definitions. The common definitions will be provided in all WSDL documents.

12.2.1 Namespaces for Parlay X Web Services

For Parlay X Web Services, the scheme is 'http', domain is www.csapi.org, and root is parlayx. Thus for XML Schema namespaces 'http://www.csapi.org/schema/parlayx' is the base name, and for WSDL 'http://www.csapi.org/wsdl/parlayx' is the base name.

12.2.2 Use of Namespaces

Correct use of namespaces is essential for both creating WSDL that will be usable by a variety of tools, and creating references that allow use of reusable content across the set of documents for a Web Service.

The following are the key namespaces defined,

- XML Schema namespaces for data type definitions
- Shared fault namespaces, for easy sharing of common fault definitions
- WSDL interface namespace for Web Service interface definitions
- WSDL schema local interface namespace for XML Schema definitions contained in the WSDL interface definition
- WSDL binding namespace for service bindings definitions

Each namespace has a distinct role. Managing them in a consistent way provides highly flexible definitions, while ensuring easy use by WSDL creators and readers.

12.2.3 Namespace Elements

The namespace definition includes three defined elements – the hierarchical name element, the version element and the namespace type element.

The hierarchical name element provides a fully qualified name in a hierarchical form for the namespace. This element is the Web Service specific information.

If a namespace contains a version number it will be a separate namespace element, immediately following the hierarchical name element, and preceding the namespace type. Version numbers are recommended, and are used in the examples.

A version number is based on release numbering, consisting of the lowercase letter 'v', followed by a number indicating major version number, followed by an underscore '_', followed by a minor version number. Any numbering beyond the minor version number follows the same convention with an underscore separator. Numbers are not limited to single digits.

Following the version number is the namespace type, which is always the last element in the namespace. The namespace type is one of 'faults' for *Shared Faults Documents*, 'interface' or 'local' for *Service Interface Documents*, or 'service' for *Service Bindings Documents*. The *Type Definitions Document* does not have a namespace type; since it does not share its namespace 'schema' (XML Schema definitions in the *Service Interface Document* use the 'local' namespace type).

12.2.4 Common Namespaces

Each document type has some common namespaces will be used in every instance of that document type.

Type Definition Documents

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Shared Faults Documents and *Service Interface Documents*

```
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Service Bindings Documents for SOAP over HTTP

```
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Other bindings will have other namespace definitions that will be common to all *Service Bindings Documents* using that binding.

12.2.5 Target Namespace

The target namespace defines the namespace that is the default namespace for the elements within a document. A special case is the target namespace defined for the schema section within the wsdl:types section of a *Service Interface Document*, where the target namespace applies specifically to the XML Schema definitions within this section.

The base namespace for WSDL related elements is http://www.example.com/wsdl. For sub-namespaces, they will extend this namespace. All elements defined within the root namespace are defined within this base target namespace.

For example target namespaces may include,

Root namespace, http://www.example.com/wsdl/v1_0/service

Sub-namespace, http://www.example.com/wsdl/accounts/v1_0/service

Multi-level sub-namespace, http://www.example.com/wsdl/accounts/payables/v1_0/service

The target namespace is the same namespace that will be defined later as the XML Schema or WSDL namespace.

12.2.6 WSDL and Schema Namespaces

Namespaces are defined for the WSDL and Schema elements that are defined within this document, and for those that are referenced by elements in this document. For each instance, a pair of namespaces may be defined (if applicable). The WSDL namespace is defined with its *Short Name*. The Schema namespace is defined with the *Short Name* plus the suffix '_xsd'.

For the WSDL reference used for this document, the name space definition is the same as the targetNamespace. For the Schema reference, the base namespace is http://www.example.com/schema, with the same hierarchy reference following the base namespace, but without an ending qualifier since the schema namespace is not shared across documents.

Examples,

Base namespace

```
xmlns:example="http://www.example.com/wsdl/v1_0"
xmlns:example_xsd="http://www.example.com/schema/v1_0"
```

Sub-namespace

```
xmlns:accounts="http://www.example.com/wsdl/accounts/v1_0/service"
xmlns:accounts_xsd="http://www.example.com/schema/accounts/v1_0"
```

12.2.7 Local Namespace Use

Within the WSDL service definition, XML Schema is used to define messages. These are defined within the wsdl:types section of the *Service Interface Document*. Since namespaces must be unique across documents, and within different

sections of the same document, a local namespace is used for the XML Schema types defined within the wsdl:types section.

The local namespace definition within a *Service Interface Document* uses the 'schema' namespace, with the *Base Name* and *Short Name* elements followed by the version element and '/local'. The namespace is defined as the *Short Name* plus '_local_xsd'. This approach guarantees unique and predictable name use.

12.2.8 Examples

Base definitions,

```
<definitions
   name="example"
   targetNamespace="http://www.example.com/wsdl/v1_0/service"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:example="http://www.example.com/wsdl/v1_0/service"
   xmlns:example_xsd="http://www.example.com/schema/v1_0">
```

Sub-namespace definitions,

```
<definitions
   name="accounts"
   targetNamespace="http://www.example.com/wsdl/accounts/v1_0/service"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:example="http://www.example.com/wsdl/v1_0"
   xmlns:example_xsd="http://www.example.com/schema/v1_0"
   xmlns:accounts="http://www.example.com/wsdl/accounts/v1_0/service"
   xmlns:accounts_xsd="http://www.example.com/schema/accounts/v1_0">
```

12.3 Authoring Style – Document Content and Names

12.3.1 General WSDL Document Information

The following are general guidelines for WSDL document information

- WSDL documents will use UTF-8 as their encoding. UTF-16 may also be used if required.
- A date in a comment at the top of the WSDL document will indicate the last revision date of the definition.

12.3.2 Names

Names will be normal language names, without prefixes (e.g. type or interface markers). The names will be meaningful, and not abbreviated in a way that makes the name hard to understand for users of the WSDL that are not literate in computer programming.

As a guideline, a person using the WSDL will be able to load the WSDL file into an XML viewer and see the names displayed and have reasonable understanding of the content.

This does not preclude the use of commonly understood acronyms within names (e.g. ATM) or commonly used abbreviations (e.g. max). However, the resulting name must still be meaningful.

12.3.3 Case Usage for Names

Two general cases are provided for, both using mixed case names; one with a leading capital letter, the other with a leading lowercase letter.

Names for all elements (all cases where the text name='Name' is used) will start with a letter and be mixed case, with the leading letter of each word capitalized. Words will not be separated by white space, underscore, hyphen or other non-letter character.

The following elements will have a leading uppercase letter – simpleType name, complexType name, interface (portType) name, binding name, service name, union element name.

The following elements will have a leading lowercase letter – field names (those names used for elements within other elements), message name (message name portion, service prefix will have uppercase letter if used), message part name, interface operation name, binding operation name.

For example, valid names include 'Name', 'FirstName', 'Name1', 'mixedCaseName'. Invalid names include '1Name', 'NAME', 'nAME'.

12.3.4 Naming Conventions for Special Names

Some names have special meaning, and are often recognized by a naming convention. For example, in some conventions constants are identified by using all upper case letters and underscores between words.

In WSDL, the case usage for names will be followed as described in the previous section. No other conventions for case usage will be used.

For faults, the fault name will be suffixed with the word 'Exception'.

In many technologies, the return value of an operation is not named. However, in WSDL the response message contains a named part. The part representing the response message content will use the name 'result'.

12.3.5 Document Layout

To provide easy and consistent reading of WSDL files, the following layout patterns are recommended.

Each tag level is indented one level relative to the previous tag indent level. The xml tag, date comment and root tag are not indented, they are on the left margin.

Indents of 3 spaces are used, and tabs are not used for storage (store files with spaces).

Namespaces are defined one per line, single spaced, indented one indent level.

Import statements are defined one per line, single spaced, with attributes on the same line.

Each primary element within the schema is separated by one blank line.

Each element within a primary element is single spaced.

Restrictions, extensions and elements are defined on a single line with their attributes.

XML Schema types are laid out according to their respective sections in this document.

Messages are defined single spaced, with one blank line separating each message definition. Messages with no parts are defined with one tag.

Interfaces (portTypes) are defined with its attributes on a single line, with one blank line separating each interface definition. The first operation starts on the line following the interface definition, with each operation defined single spaced and with a blank line separating each operation definition. Each element defined within an operation (input, output and fault) is defined on a single line with its attributes within one tag.

Bindings will be laid out consistently with interfaces.

Each service is defined single spaced.

12.4 Data Type Definitions

All data type definition examples are shown using XML Schema.

12.4.1 Types Section Declaration

All data types are defined using XML Schema in the *Type Definitions Document*.

Base document definition,

```
<xsd:schema>
   targetNamespace=”http://www.example.com/schema/v1_0”
   xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>

   <!-- types -->
</xsd:schema>
```

Sub-namespace document definition,

```
<xsd:schema>
   targetNamespace=“http://www.example.com/schema/accounts/v1_0“
   xmlns:xsd=“http://www.w3.org/2001/XMLSchema“>

   <!-- types -->
</xsd:schema>
```

12.4.1.1 Optional Elements

XML Schema allows elements to be defined as optional, meaning that the element may or may not be present within an XML document. Elements may be declared as optional by use of the minOccurs attribute with a value of zero.

```
<xsd:element name=”example” type=”xsd:int” minOccurs=”0” maxOccurs=”1”/>
```

12.4.1.2 Nillable Elements

XML Schema allows elements to be declared as nillable, indicating that the value may be nil (unspecified, not just a zero value).

<xsd:element name=”example” type=”xsd:int” nillable=”true” />

12.4.1.3 User Defined Simple Data Types

User defined simple data types are defined as XML Schema simpleType elements with a name attribute and a restriction to the base XML Schema type. Simple data types do not extend other simple data types.

```
<xsd:simpleType name=”Example”>
    <xsd:restriction base=”xsd:int”/>
</xsd:simpleType>
```

12.4.1.4 Data Structures

Data structures are defined as complex types with a sequence of elements within the complex type.

```
<xsd:complexType name=”Account”>
   <xsd:sequence>
      <xsd:element name=”name” type=”xsd:string”/>
      <xsd:element name=”address” type=”example_xsd:Address”/>
   </xsd:sequence>
</xsd:complexType>
```

12.4.1.5 Enumerations

Enumerations are defined using XML Schema enumerations. It is suggested that the values listed in an enumeration follow the mixed case usage (Initial capital, capital for each word, lowercase otherwise) and always start with a letter.

This will ensure compatibility with programming language usage, and provide consistency in readability of enumerations.

```
<xsd:simpleType name="Days">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Sunday"/>
      <xsd:enumeration value="Monday"/>
      <xsd:enumeration value="Tuesday"/>
   </xsd:restriction>
</xsd:simpleType>
```

Enumerations are assigned the literal values from the list provided, not a generated integer representing each enumeration values.

12.4.1.6 Unions

Unions allow a single data type to have one of a set of values. This is a constructed data type, consisting of an enumeration element that indicates which value is present and the value itself. The enumeration provides a list of possible values, and the union element provides a value that corresponds with each member of the enumeration.

```
<xsd:simpleType name="PhoneType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Home"/>
      <xsd:enumeration value="Work"/>
      <xsd:enumeration value="Mobile"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="PhoneNumber">
   <xsd:sequence>
      <xsd:element name="UnionElement" type="example_xsd:PhoneType">
      <xsd:element name="Home" type="xsd:string"/>
      <xsd:element name="Work" type="xsd:string"/>
      <xsd:element name="Mobile" type="example_xsd:MobileNumber"/>
   </xsd:sequence>
</xsd:complexType>
```

12.4.1.7 Web Service References

WSDL does not define a data type for a reference to a Web Service; however a data type can be produced that will provide the information necessary for another system to invoke a Web Service. This equivalent is a Web Service reference. This is a precursor to the use of WS-Addressing to provide this capability.

The Web Service reference consists of three parts:

- The URI representing the end point, this is equivalent to the value populated in the <service> section of a WSDL definition.
- The port type in the reference allows the entity that will be the Web Service Requester to determine which port type or client stub to use.
- To support stateful references, the additional data element, correlator, is provided. The correlator is opaque to the service receiving the reference; it is meaningful only to the system on which the Web Service is invoked. If a service is stateless, this item will be empty.

```
<xsd:complexType name="SimpleReference">
   <xsd:sequence>
      <xsd:element name="endpoint" type="xsd:anyURI"/>
      <xsd:element name="portType" type="xsd:string"/>
      <xsd:element name="correlator" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Any additional information necessary required to access the Web Service referred to by a reference, e.g. security content, must be provided by the implementation.

12.5 Messages and Interfaces (PortTypes)

12.5.1 Messages

Messages are used in the operation elements of interfaces (portTypes), providing the definition of the content that is exchanged on input, output and faults.

12.5.1.1 Document Style Web Services

Document style Web Services define one input message and one output message, each with one part that references an element defined with XML Schema. These may be combined with fault messages in the definition of operations within an interface.

The XML Schema elements that define the message parts are defined within the wsdl:types section of the *Service Interface Document*. These parts may include references to data types defined in *Type Definition Documents*. Faults specific to the interface defined may also have their messages defined in this manner.

12.5.2 Interfaces (PortTypes)

Interfaces make a set of operations available, and define the messages that will be used for each.

For the request / response message pattern, an interface will have an operation definition that contains a single input message, a single output message and zero or more fault messages, in that order.

12.5.3 Faults (Exceptions)

There are four common types of faults that may be part of interface definitions,

1. SOAP faults, that occur before a message is received by the Web Service
2. Service faults, that are generated as a result of a system failure, resource failure or rejection of the message (e.g. invalid message content)
3. Policy faults, that are the result of the provider of the Web Service rejecting the request, due to a reason other than those covered by a service fault, and not specific to a service (e.g. privacy)
4. Service specific faults, that represent a fault that is not common across services

In defining interfaces, these faults are represented using the following approach,

- SOAP faults are not defined in the WSDL, their content is defined independently. Usually these faults are generated by intermediaries or as part of the infrastructure (e.g. security subsystem).
- Every operation shall include a ServiceException, providing a common manner in which these faults can be provided back to the requester.
- Every operation shall include a PolicyException, providing a common manner in which these faults can be provided back to the requester.
- Only faults that fall outside the service and policy faults should be provided additional fault definitions – in many cases, no additional fault definitions are required.

By following these guidelines, the following desirable characteristics of Web Services will be provided,

- Applications, Web Services, and intermediaries can implement consistent handling of classes of faults without specific knowledge of the particular Web Service implementation.
- Faults can be minimized, allowing service specific faults to be clearly recognized, not cluttered amongst many other common faults.

Even when a Web Service does not initially require a ServiceException or PolicyException, many will be deployed in different places or with additional requirements over time, and will likely require one or both of these over time. Providing these initially reduces the impact of change over time and use.

Also, combining the various service and policy faults into these common fault definitions allows flexible use of the content of the two fault definitions, allowing extensibility over time without impacting the service definition.

12.6 Bindings and Service Definitions

12.6.1 Binding

The binding defines how the WSDL definitions will be utilized in interacting with the network. The binding defines the protocols and operational style of the binding. For example, SOAP over HTTP or SOAP over SMTP for protocols, and document or rpc for style.

While the binding is specific to a technology, unlike the other parts of the WSDL documents, the binding does have influence on the overall service definition. For instance, the choice of SOAP style affects how messages are defined, and the binding choices may determine semantics related to the implementation – for example, some bindings may have limitations in support for asynchronous or reliable messaging.

This document does not address bindings in detail, as the binding is independent of the WSDL interface definitions. Specific information on use of the SOAP/HTTP binding is covered in the WS-I Basic Profile [8].

12.6.2 Service Definition

Services define an endpoint (port), though the address of the endpoint specified in this definition is often replaced at runtime when the discovery step determines that actual location that the service is hosted at.

During development, it is reasonable to use a location that corresponds to a debugging location at which the service will be tested – often pointing to localhost and containing default URL information for a development configuration to be used for testing. For deployment, the location should be a default location for service access.

```
<wsdl:service name=”MessageService”>
   <wsdl:port binding=”msg:MessageServiceBinding” name=”MessageService”>
         <soap:address location=”http://www.example.com/services/MessageService”/>
   </wsdl:port>
</wsdl:service>
```

At runtime, the Web Service Application can determine the soap:address location information through local configuration or through a discovery process, replacing the location information in the default service definition.

Annex A (normative): WSDL for Common Data Definitions

The document/literal WSDL representation of this interface specification is compliant to the content requirements specified in this document and is contained in text files (contained in archive 29199-01-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-2 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 2: Third Party Call
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 5
2 References 5
3 Definitions and Abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 6
4 Detailed Service Description 6
5 Namespaces 7
6 Sequence Diagrams 7
6.1 'Click to Dial' Call Setup 7
7 XML Schema Data Type Definition 8
7.1 CallStatus Enumeration 8
7.2 CallTerminationCause Enumeration 8
7.3 CallInformation Structure 9
8 Web Service Interface Definition 9
8.1 Interface : ThirdPartyCall 9
8.1.1 Operation : MakeCall 9
8.1.1.1 Input message : MakeCallRequest 9
8.1.1.2 Output message : MakeCallResponse 9
8.1.1.3 Referenced Faults 10
8.1.2 Operation : GetCallInformation 10
8.1.2.1 Input message : GetCallInformationRequest 10
8.1.2.2 Output message : GetCallInformationResponse 10
8.1.2.3 Referenced Faults 10
8.1.3 Operation : EndCall 10
8.1.3.1 Input message : EndCallRequest 10
8.1.3.2 Output message : EndCallResponse 10
8.1.3.3 Referenced Faults 11
8.1.4 Operation : CancelCall 11
8.1.4.1 Input message : CancelCallRequest 11
8.1.4.2 Output message : CancelCallResponse 11
8.1.4.3 Referenced Faults 11
9 Fault Definitions 11
9.1 ServiceException 11
9.1.1 SVC0260: Call already connected 11
9.1.2 SVC0261: Call already terminated 12
10 Service Policies 12
Annex A (normative): WSDL for Third Party Call 13
Annex B (informative): Change history 14

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 2 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	**Third Party Call**
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 2 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Third Party Call Web Service aspects of the interface. All aspects of the Third Party Call Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

Currently, in order to perform a third party call in telecommunication networks we have to write applications using specific protocols to access Call Control functions provided by network elements (specifically operations to initiate a call from applications). This approach requires a high degree of network expertise. We can also use the OSA gateway approach, invoking standard interfaces to gain access to call control capabilities, but these interfaces are usually perceived to be quite complex by application IT developers. Developers must have advanced telecommunication skills to use Call Control OSA interfaces.

In this subclause we describe a Parlay X Web Service, Third Party Call, for creating and managing a call initiated by an application (third party call). The overall scope of this Web Service is to provide functions to application developers to create a call in a simple way. Using the Third Party Call Web Service, application developers can invoke call handling functions without detailed telecommunication knowledge.

Figure 1 shows an scenario using the Third Party Call Web Service to handle third party call functions. The application invokes a web service to retrieve stock quotes and a Parlay X Interface to initiate a third party call between a broker and his client.

In the scenario, whenever a particular stock quote reaches a threshold value (1) and (2), the client application invokes a third party call between one or more brokers and their corresponding customers to decide actions to be taken. After invocation (3) by the application, the Third Party Call Web Service invokes a Parlay API method (4) using the Parlay/OSA SCS-CC (Call control) interface. This SCS handles the invocation and sends a message (5) to an MSC to set-up a call between user A and user B.

In an alternative scenario, the Parlay API interaction involving steps (4) and (5) could be replaced with a direct interaction between the Third Party Call Web Service and the Mobile network.

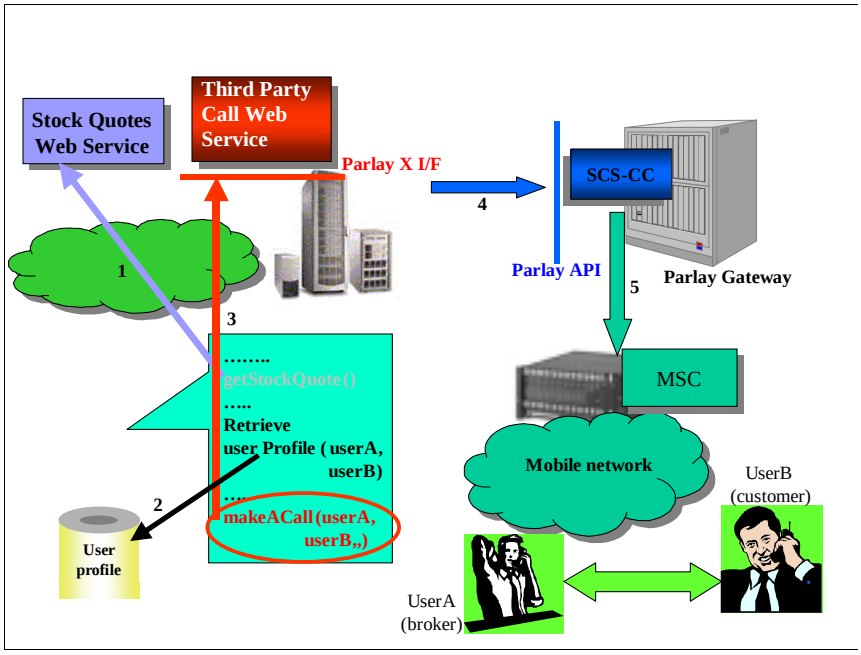


Figure 1: Third Party Call Scenario

5 Namespaces

The ThirdPartyCall interface uses the namespace

www.csapi.org/wsdl/parlayx/third_party_call/v2_0

The data types are defined in the namespace

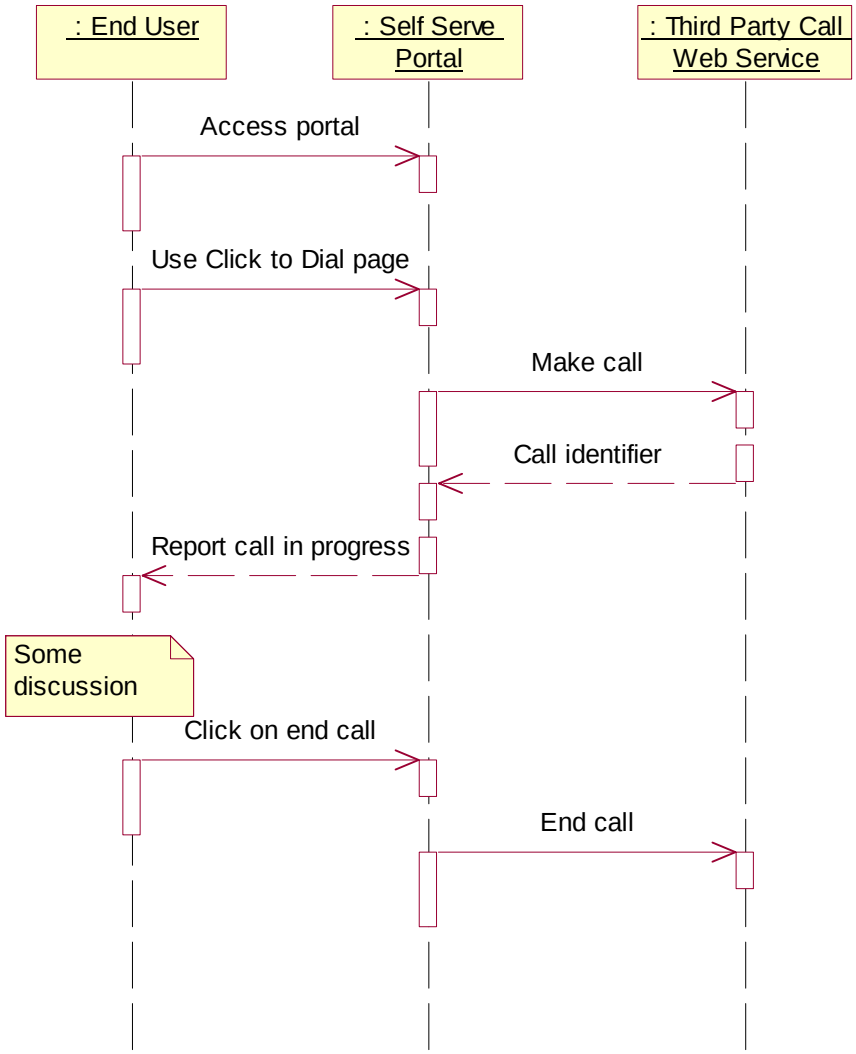
www.csapi.org/schema/parlayx/ third_party_call /v2_0

The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 ‘Click to Dial’ Call Setup

A common convergence application is Click to Dial, where a self service portal provides a web page that can initiate a call between two phones. This sequence shows a basic call setup, and ending the call through the portal.



7 XML Schema Data Type Definition

7.1 CallStatus Enumeration

List of call status values.

Enumeration	Description
CallInitial	The call is being established
CallConnected	The call is active
CallTerminated	The call was terminated

7.2 CallTerminationCause Enumeration

List of call termination cause values.

Enumeration	Description

CallingPartyNoAnswer	Calling Party did not answer
CalledPartyNoAnswer	Called Party did not answer
CallingPartyBusy	Calling Party was busy
CalledPartyBusy	Called Party was busy
CallingPartyNotReachable	Calling Party was not reachable
CalledPartyNotReachable	Called Party was not reachable
CallHangUp	The call was terminated by either party hanging up
CallAborted	The call was aborted (any other termination cause)

7.3 CallInformation Structure

Call information for this call.

Element Name	Element Type	Description
CallStatus	CallStatus	It indicates the current status of the call (see possible values below)
StartTime	xsd:dateTime	When applicable (callStatus <> CallInitial), it indicates the time of the beginning of the call
Duration	xsd:int	When applicable (callStatus = CallTerminated), it indicates the duration of the call expressed in seconds
TerminationCause	CallTerminationCause	When applicable (callStatus = CallTerminated), it indicates the cause of the termination of the call

8 Web Service Interface Definition

8.1 Interface : ThirdPartyCall

This interface provides the ability to setup, end and determine the status of a call.

8.1.1 Operation : MakeCall

The invocation of **MakeCall** requests to set-up a voice call between two addresses, **CallingParty** and **CalledParty**, provided that the invoking application is allowed to connect them. Optionally the application can also indicate the charging information (**Charging**).

By invoking this operation the application may monitor the status of the requested call. The returned parameter, **CallIdentifier**, can be used to identify the call. In order to receive the information on call status the application has to explicitly invoke **GetCallInformation**.

8.1.1.1 Input message : MakeCallRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the first user involved in the call
CalledParty	xsd:anyURI	It contains the address of the second user involved in the call
Charging	common:ChargingInformation	Charge to apply to the call (optional)

8.1.1.2 Output message : MakeCallResponse

Part Name	Part Type	Description

CallIdentifier	xsd:string	It identifies a specific call request

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error
- POL0008 – Charging not supported

8.1.2 Operation : GetCallInformation

The invocation of **GetCallInformation** retrieves the current status, **CallInformation**, of the call identified by **CallIdentifier**. This method can be invoked multiple times by the application even if the call has already ended. However, after the call has ended, status information will be available only for a limited period of time that is specified in the service policy 'StatusRetentionTime'.

8.1.2.1 Input message : GetCallInformationRequest

Part Name	Part Type	Description
CallIdentifier	String	It identifies a specific call request

8.1.2.2 Output message : GetCallInformationResponse

Part Name	Part Type	Description
CallInformation	CallInformation	It identifies the status of the call

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.1.3 Operation : EndCall

The invocation of **EndCall** terminates the call identified by **CallIdentifier**. If the call is still in the initial state this method has the same effect as the **CancelCallRequest** method.

8.1.3.1 Input message : EndCallRequest

Part Name	Part Type	Description
CallIdentifier	String	It identifies a specific call request

8.1.3.2 Output message : EndCallResponse

Part Name	Part Type	Description

None		

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0261 – Call already terminated

PolicyException from [6]

- POL0001 – Policy error

8.1.4 Operation : CancelCall

The invocation of **CancelCallRequest** cancels the previously requested call identified by **CallIdentifier**. Note that this method differs from the **EndCall** method since it only attempts to prevent the call from starting but it does not have any effect if the call has already started.

8.1.4.1 Input message : CancelCallRequest

Part Name	Part Type	Description
CallIdentifier	String	It identifies a specific call request

8.1.4.2 Output message : CancelCallResponse

Part Name	Part Type	Description
None		

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0260 – Call already connected

PolicyException from [6]

- POL0001 – Policy error

9 Fault Definitions

The following faults are defined for this service.

9.1 ServiceException

9.1.1 SVC0260: Call already connected

Message Id	SVC0260

Text	Call has already been connected, it cannot be canceled.
Variables	None

9.1.2 SVC0261: Call already terminated

Message Id	SVC0261
Text	Call has already been terminated.
Variables	None.

10 Service Policies

These service policies are defined for the Third Party Call service.

Name	Type	Description
ChargingAllowed	xsd:boolean	Is charging allowed for makeCall operation.
StatusRetentionTime	xsd:int	Length of time, in seconds, to retain status after the termination of the call.

Annex A (normative): WSDL for Third Party Call

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-02-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-3 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 3: Call Notification
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 7
6 Sequence Diagrams 7
6.1 SMS Notification of a Missed Call 7
7 XML Schema Data Type Definition 8
7.1 ActionValues Enumeration 8
7.2 Action Structure 8
8 Web Service Interface Definition 8
8.1 Interface : CallDirection 8
8.1.1 Operation : HandleBusy 9
8.1.1.1 Input message : handleBusyRequest 9
8.1.1.2 Output message : handleBusyResponse 9
8.1.1.3 Referenced Faults 9
8.1.2 Operation : HandleNotReachable 9
8.1.2.1 Input message : handleNotReachableRequest 10
8.1.2.2 Output message : handleNotReachableResponse 10
8.1.2.3 Referenced Faults 10
8.1.3 Operation : HandleNoAnswer 10
8.1.3.1 Input message : handleNoAnswerRequest 10
8.1.3.2 Output message : handleNoAnswerResponse 10
8.1.3.3 Referenced Faults 10
8.1.4 Operation : HandleCalledNumber 11
8.1.4.1 Input message : handleCalledNumberRequest 11
8.1.4.2 Output message : handleCalledNumberResponse 11
8.1.4.3 Referenced Faults 11
8.2 Interface : CallNotification 11
8.2.1 Operation : NotifyBusy 11
8.2.1.1 Input message : NotifyBusyRequest 11
8.2.1.2 Output message : NotifyBusyResponse 11
8.2.1.3 Referenced Faults 12
8.2.2 Operation : NotifyNotReachable 12
8.2.2.1 Input message : NotifyNotReachableRequest 12
8.2.2.2 Output message : NotifyNotReachableResponse 12
8.2.2.3 Referenced Faults 12
8.2.3 Operation : NotifyNoAnswer 12
8.2.3.1 Input message : NotifyNoAnswerRequest 12
8.2.3.2 Output message : NotifyNoAnswerResponse 12
8.2.3.3 Referenced Faults 12
8.2.4 Operation : NotifyCalledNumber 12
8.2.4.1 Input message : NotifyCalledNumberRequest 13
8.2.4.2 Output message : NotifyCalledNumberResponse 13
8.2.4.3 Referenced Faults 13

9 Fault Definitions 13
10 Service Policies 13
Annex A (normative): WSDL for Call Notification 14
Annex B (informative): Change history 15

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 3 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	**Call Notification**
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 3 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Call Notification Web Service aspects of the interface. All aspects of the Call Notification Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

Currently, in order to determine the handling of a subscriber initiated call in telecommunication networks we have to write applications using specific protocols to access Call Control functions provided by network elements. This approach requires a high degree of network expertise. We can also use the OSA gateway approach, invoking standard interfaces to gain access to call control capabilities, but these interfaces are usually perceived to be quite complex by application IT developers. Developers must have advanced telecommunication skills to use Call Control OSA interfaces.

In this subclause we will describe a Parlay X Web Service, Call Notification, for handling calls initiated by a subscriber in the network. A (third party) application determines how the call should be treated. The overall scope of this Web Service is to provide simple functions to application developers to determine how a call should be treated. Using the Web Service, application developers can perform simple handling of network-initiated calls without specific Telco knowledge.

Examples of usage include the following.

Incoming call handling. A subscriber receives a call while he is logged-on to the Internet. Since this occupies his telephone connection, he is regarded as busy by the network. The subscriber has an application that is invoked when somebody tries to call him while he is busy. The application provides the subscriber with a list of choices on how to handle the call (e.g., route the call to voicemail, redirect the call to a secretary, reject the call). Based on the response of the subscriber the call is handled in the network. Alternatively, the call is re-routed or released depending on the preferences of the subscriber and some context information (e.g., based on the status or location of the subscriber).

Service numbers. An application is triggered whenever a certain service number is dialled. This number is used to connect the caller to one of the maintenance personnel. The application redirects the call to the appropriate maintenance person based on, e.g., calling party number, time, location and availability of the maintenance personnel.

SMS notification of missed calls. An application offers the subscriber the possibility to be notified via SMS whenever he misses a call. The application registers to be notified when calls to its subscribers encounter busy, no-answer or not-reachable. The application does not influence the call treatment, but sends an SMS containing the calling party number, the time and reason why the call was missed.

5 Namespaces

The Call Notification interface uses the namespace

www.csapi.org/wsdl/parlayx/call_notification/v2_0

The data types are defined in the namespace

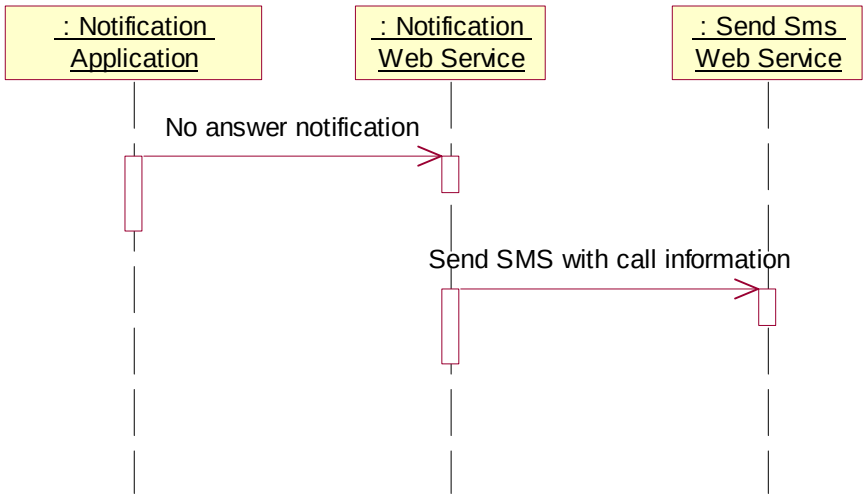
www.csapi.org/schema/parlayx/call_notification/v2_0

The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 SMS Notification of a Missed Call

Showing the use of the CallNotification and SendSms services, an SMS is sent to a person who misses a call (no answer). This sequence assumes that the provisioning of the call notification has occurred independently.



7 XML Schema Data Type Definition

7.1 ActionValues Enumeration

The **ActionValues** data type is an enumeration with the following values:

Enumeration	Description
Route	Request to (re-)route the call to the address indicated with routingAddress.
Continue	Request to continue the call without any changes. This will result in normal handling of the event in the network
EndCall	Request to end the call. This will result in termination of the call. The callingParty will receive a tone or announcement.

7.2 Action Structure

The **Action** data type is a structure containing the following parameters:

Element Name	Element Type	Description
ActionToPerform	ActionValues	Indicates the action as described below
RoutingAddress	xsd:anyURI	The address to be used in case the action indicates 'Route'
Charging	common:ChargingInformation	Charge to apply to this call

8 Web Service Interface Definition

8.1 Interface : CallDirection

This subclause describes an initial set of capabilities in terms of message invocations, parameters and data types. The message-based invocations are:

- handleBusy

- handleNotReachable
- handleNoAnswer
- handleCalledNumber

These messages are initiated by the Call Notification Web Service (running in a Parlay X Gateway) and invoke an application web service(s), as a result of activity in the network. The result of the invocation of a handle<Event> operation is used as an indication on how the call should be handled in the network. The application can not keep control over the call after handling the event; every event handling is a separate occurrence.

Note that because the results of the invocations of the application web service(s) determine call handling in the network, the names of the methods are prefixed with 'handle', rather than 'notify'. The prefix 'notify' would imply a more asynchronous behaviour, whereas 'handle' shows the synchronous nature of these invocations.

The criteria for which the application web service(s) should be invoked, such as type of events (busy, answer etc.), a URI to the Web Service and triggered addresses should be provisioned by the operator in an off-line process.

8.1.1 Operation : HandleBusy

The invocation of **handleBusy** requests the application to inform the gateway how to handle the call between two addresses, the **callingParty** and the **calledParty**, where the **calledParty** is busy when the call is received. The application returns the **action**, which directs the gateway to perform one of the following actions:

- "Continue", resulting in normal handling of the busy event in the network, e.g. playing of a busy tone to the **callingParty**
- "EndCall", resulting in the call being terminated; the exact tone or announcement that will be played to the **callingParty** is operator-specific
- "Route", resulting in the call being re-routed to a **calledParty** specified by the application.

Optionally, in the **action** parameter, the application can also indicate the charging information.

8.1.1.1 Input message : handleBusyRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party is busy.

8.1.1.2 Output message : handleBusyResponse

Part Name	Part Type	Description
Action	Action	It indicates the action to be performed by the gateway.

8.1.1.3 Referenced Faults

None.

8.1.2 Operation : HandleNotReachable

The invocation of **handleNotReachable** requests the application to inform the gateway how to handle the call between two addresses, the **callingParty** and the **calledParty**, where the **calledParty** is not reachable when the call is received. The application returns the **action**, which directs the gateway to perform one of the following actions:

- "Continue", resulting in normal handling of the 'not reachable' event in the network, e.g. playing of a busy tone to the **callingParty**

- "EndCall", resulting in the call being terminated; the exact tone or announcement that will be played to the **callingParty** is operator-specific
- "Route", resulting in the call being re-routed to a **calledParty** specified by the application.

Optionally, in the **action** parameter, the application can also indicate the charging information.

8.1.2.1 Input message : handleNotReachableRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party is not reachable.

8.1.2.2 Output message : handleNotReachableResponse

Part Name	Part Type	Description
Action	Action	It indicates the action to be performed by the gateway.

8.1.2.3 Referenced Faults

None.

8.1.3 Operation : HandleNoAnswer

The invocation of **handleNoAnswer** requests the application to inform the gateway how to handle the call between two addresses, the **callingParty** and the **calledParty**, where the **calledParty** does not answer the received call. The application returns the **action**, which directs the gateway to perform one of the following actions:

- "Continue", resulting in normal handling of the 'no answer' event in the network, e.g. playing of a busy tone to the **callingParty**
- "EndCall", resulting in the call being terminated; the exact tone or announcement that will be played to the **callingParty** is operator-specific
- "Route", resulting in the call being re-routed to a **calledParty** specified by the application.

Optionally, in the **action** parameter, the application can also indicate the charging information.

8.1.3.1 Input message : handleNoAnswerRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party does not answer the call.

8.1.3.2 Output message : handleNoAnswerResponse

Part Name	Part Type	Description
Action	Action	It indicates the action to be performed by the gateway.

8.1.3.3 Referenced Faults

None.

8.1.4 Operation : HandleCalledNumber

The invocation of **handleCalledNumber** requests the application to inform the gateway how to handle the call between two addresses, the **callingParty** and the **calledParty**. The method is invoked when the **callingParty** tries to call the **calledParty**, but before the network routes the call to the **calledParty**. For example, the **calledParty** does not have to refer to a real end user, i.e., it could be a service number. The application returns the **action**, which directs the gateway to perform one of the following actions:

- "Continue", resulting in normal handling in the network, i.e. the call will be routed to the **calledParty** number, as originally dialled
- "EndCall", resulting in the call being terminated; the exact tone or announcement that will be played to the **callingParty** is operator-specific
- "Route", resulting in the call being re-routed to a **calledParty** specified by the application.

Optionally, in the **action** parameter, the application can also indicate the charging information.

8.1.4.1 Input message : handleCalledNumberRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party.

8.1.4.2 Output message : handleCalledNumberResponse

Part Name	Part Type	Description
Action	Action	It indicates the action to be performed by the gateway.

8.1.4.3 Referenced Faults

None.

8.2 Interface : CallNotification

When call events occur in the network, the application may be notified of these events. The application does not have the ability to influence the call, as call processing continues.

Notifications are provided for call attempt, busy, not reachable and no answer events.

8.2.1 Operation : NotifyBusy

A busy notification informs the application that a call between two parties was attempted, but the called party was busy.

8.2.1.1 Input message : NotifyBusyRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party is busy.

8.2.1.2 Output message : NotifyBusyResponse

Part Name	Part Type	Description

None		

8.2.1.3 Referenced Faults

None.

8.2.2 Operation : NotifyNotReachable

A not reachable notification informs the application that a call between two parties was attempted, but the called party was not reachable.

8.2.2.1 Input message : NotifyNotReachableRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party is not reachable.

8.2.2.2 Output message : NotifyNotReachableResponse

Part Name	Part Type	Description
None		

8.2.2.3 Referenced Faults

None.

8.2.3 Operation : NotifyNoAnswer

A no answer notification informs the application that a call between two parties was attempted, but the called party did not answer.

8.2.3.1 Input message : NotifyNoAnswerRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party. This party did not answer.

8.2.3.2 Output message : NotifyNoAnswerResponse

Part Name	Part Type	Description
None		

8.2.3.3 Referenced Faults

None.

8.2.4 Operation : NotifyCalledNumber

A called number notification informs the application that a call between two parties is being attempted.

8.2.4.1 Input message : NotifyCalledNumberRequest

Part Name	Part Type	Description
CallingParty	xsd:anyURI	It contains the address of the caller.
CalledParty	xsd:anyURI	It contains the address of the called party.

8.2.4.2 Output message : NotifyCalledNumberResponse

Part Name	Part Type	Description
None		

8.2.4.3 Referenced Faults

None.

9 Fault Definitions

No new faults defined for this service.

10 Service Policies

No service policies are defined for this service.

Annex A (normative): WSDL for Call Notification

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-03-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-4 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 4: Short Messaging
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 8
6 Sequence Diagrams 9
6.1 Send SMS and Report Status 9
7 XML Schema Data Type Definition 10
7.1 DeliveryStatus Enumeration 10
7.2 SmsFormat Enumeration 10
7.3 DeliveryInformation Structure 11
7.4 SmsMessage Structure 11
8 Web Service Interface Definition 11
8.1 Interface : SendSms 11
8.1.1 Operation : SendSms 11
8.1.1.1 Input message : SendSmsRequest 11
8.1.1.2 Output message : SendSmsResponse 12
8.1.1.3 Referenced Faults 12
8.1.2 Operation : SendSmsLogo 12
8.1.2.1 Input message : SendSmsLogoRequest 12
8.1.2.2 Output message : SendSmsLogoResponse 13
8.1.2.3 Referenced Faults 13
8.1.3 Operation : SendSmsRingtone 13
8.1.3.1 Input message : SendSmsRingtoneRequest 13
8.1.3.2 Output message : SendSmsRingtoneResponse 14
8.1.3.3 Referenced Faults 14
8.1.4 Operation : GetSmsDeliveryStatus 14
8.1.4.1 Input message : GetSmsDeliveryStatusRequest 14
8.1.4.2 Output message : GetSmsDeliveryStatusResponse 15
8.1.4.3 Referenced Faults 15
8.2 Interface : SmsNotification 15
8.2.1 Operation : NotifySmsReception 15
8.2.1.1 Input message : NotifySmsReceptionRequest 15
8.2.1.2 Output message : NotifySmsReceptionResponse 15
8.2.1.3 Referenced Faults 15
8.3 Interface : ReceiveSms 15
8.3.1 Operation : GetReceivedSms 15
8.3.1.1 Input message : GetReceivedSmsRequest 16
8.3.1.2 Output message : GetReceivedSmsResponse 16
8.3.1.3 Referenced Faults 16
9 Fault Definitions 16
9.1 ServiceException 16
9.1.1 SVC0280: Message too long 16
9.1.2 SVC0281: Unrecognized data format 16
10 Service Policies 17
Annex A (normative): WSDL for Short Messaging 18
Annex B (informative): Change history 19

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 4 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	**Short Messaging**
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 4 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Short Messaging Web Service aspects of the interface. All aspects of the Short Messaging Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] and the following apply.

SMS	Short Messaging Service
SMS-C	Short Messaging Service Center

4 Detailed Service Description

Currently, in order to programmatically receive and send SMS it is necessary to write applications using specific protocols to access SMS functions provided by network elements (e.g., SMS-C). This approach requires a high degree of network expertise. Alternatively it is possible to use the Parlay/OSA approach, invoking standard interfaces (e.g., User Interaction or Messaging Service Interfaces) to gain access to SMS capabilities, but these interfaces are usually perceived to be quite complex by IT application developers. Developers must have advanced telecommunication skills to use OSA interfaces.

In this chapter we describe a Parlay X Web Service, for sending and receiving SMS messages. The overall scope of this Web Service is to provide to application developers primitives to handle SMS in a simple way. In fact, using the SMS Web Service, application developers can invoke SMS functions without specific Telco knowledge.

For sending a message to the network (see clause 4.2 of the present document, Send SMS API), the application invokes a message to send it and must subsequently become active again to poll for delivery status. There is an alternative to this polling mechanism, i.e. an asynchronous notification mechanism implemented with an application-side web service. However it was decided not to provide a notification mechanism in the first release, to make the API as simple as possible, even though the polling mechanism is not as network efficient as the notification mechanism.

For receiving a message from the network, the application may use either polling (see clause 4.4 of the present document, Receive SMS API) or notification (see clause 4.3 of the present document, SMS Notification API) mechanisms. The notification mechanism is more common: network-initiated messages are sent to autonomous application-side web services. Both mechanisms are supported, but the provisioning of the notification-related criteria is not specified.

Figure 1 shows a scenario using the SMS Web Service to send an SMS message from an application. The application invokes a web service to retrieve a weather forecast for a subscriber (1) & (2) and a Parlay X Interface (3) to use the SMS Web Service operations (i.e. to send an SMS). After invocation, the SMS Web Service invokes a Parlay API method (4) using the Parlay/OSA SCS-SMS (User Interaction) interface. This SCS handles the invocation and sends an UCP operation (5) to an SMS-C. Subsequently the weather forecast is delivered (6) to the subscriber.

In an alternative scenario, the Parlay API interaction involving steps (4) and (5) could be replaced with a direct interaction between the SMS Web Service and the Mobile network.

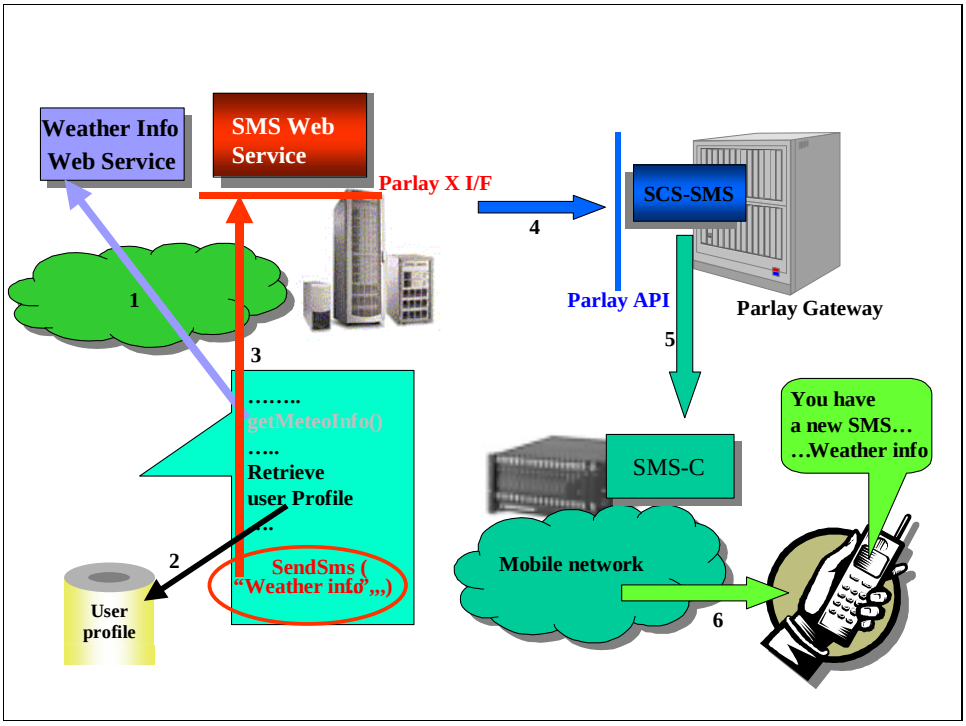


Figure 1: Send SMS Scenario

Figure 2 shows a scenario using the SMS Web Service to deliver a received SMS message to an application. The application receives a Parlay X web service invocation to retrieve an SMS sent by a subscriber (1) & (2). The SMS message contains the e-mail address of the person the user wishes to call. The application invokes a Parlay X Interface (3) to the Third Party Call Web Service in order to initiate the call (4).

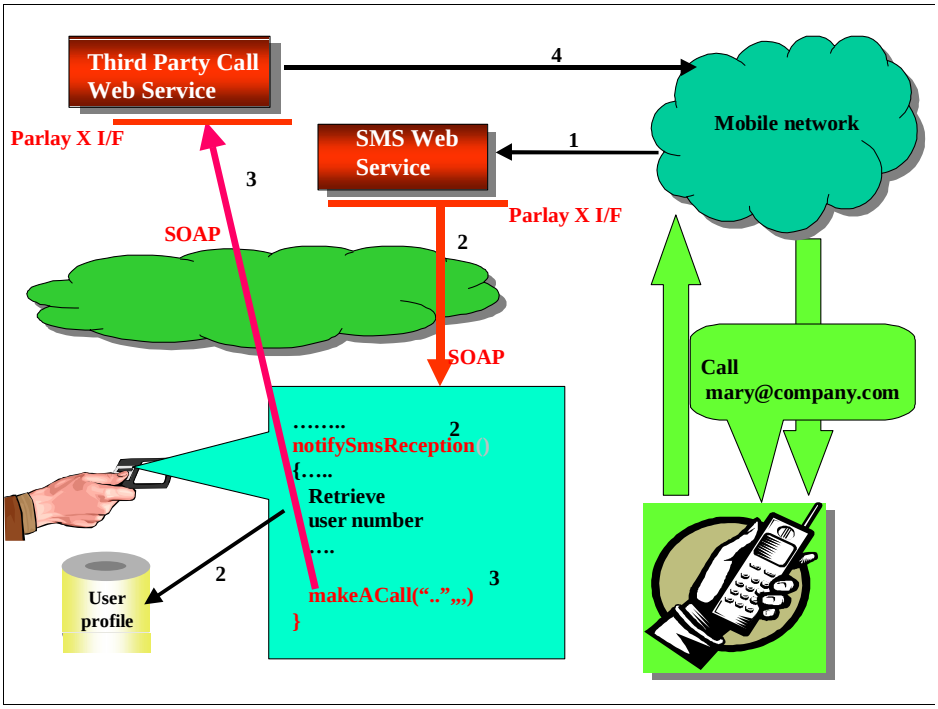


Figure 2: Receive SMS Scenario

5 Namespaces

The SendSms interface uses the namespace

www.csapi.org/wsdl/parlayx/sms/send/v2_0

The ReceiveSms interface uses the namespace

www.csapi.org/wsdl/parlayx/sms/receive/v2_0

The SmsNotification interface uses the namespace

www.csapi.org/wsdl/parlayx/sms/notification/v2_0

The data types are defined in the namespace

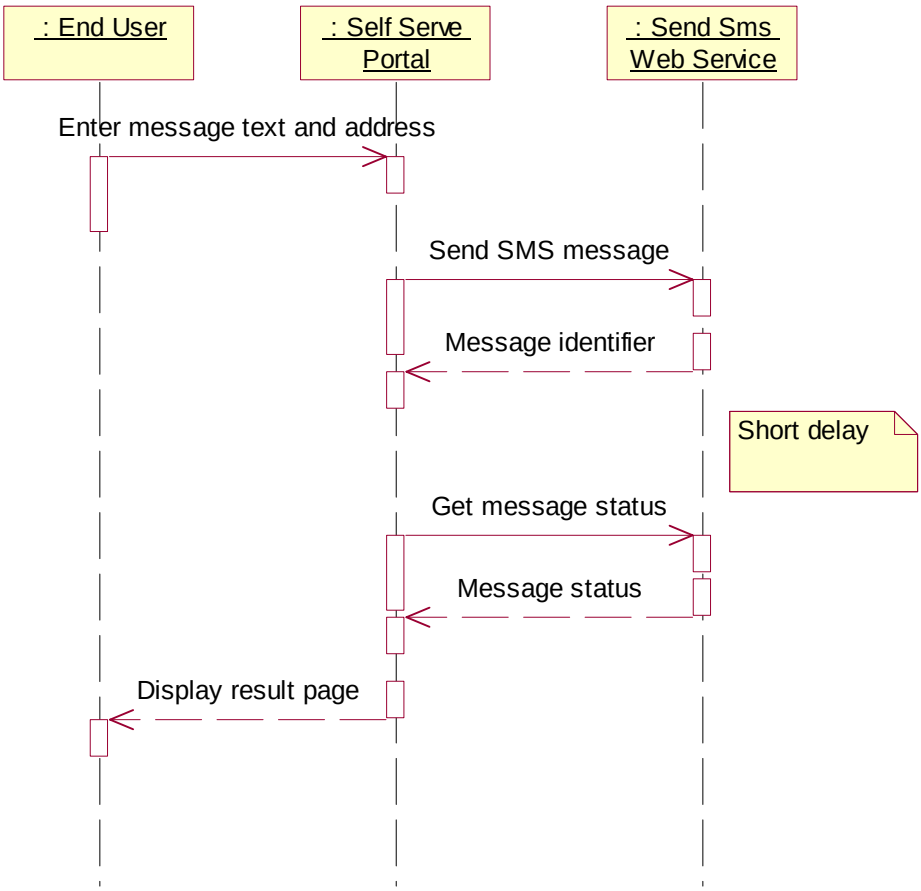
www.csapi.org/schema/parlayx/sms/v2_0

The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Send SMS and Report Status

Sending SMS message from Web portals is a common capability offered by Service Providers. This sequence diagram shows a portal providing this service.



7 XML Schema Data Type Definition

7.1 DeliveryStatus Enumeration

List of delivery status values.

Enumeration	Description
Delivered	Successful delivery
DeliveryUncertain	Delivery status unknown: e.g. because it was handed off to another network.
DeliveryImpossible	Unsuccessful delivery; the message could not be delivered before it expired.
MessageWaiting	The message is still queued for delivery. This is a temporary state, pending transition to one of the preceding states.

7.2 SmsFormat Enumeration

List of SMS format values.

Enumeration	Description
Ems	Enhanced Messaging Service, standardized in 3GPP TS 23.040 [3], which defines a logo/ringtone format

SmartMessaging™ Defines a logo/ringtone format

7.3 DeliveryInformation Structure

Delivery status information.

Element Name	Element Type	Description
Address	xsd:anyURI	It indicates the destination address to which the notification is related
DeliveryStatus	DeliveryStatus	Indicates the delivery result for destinationAddress. Possible values are: 'Delivered', 'DeliveryUncertain', 'DeliveryImpossible'.

7.4 SmsMessage Structure

SMS message information. The SenderAddress is the address from which the message was actually sent, which may or may not match the senderName value provided in the SendSms operation.

Element Name	Element Type	Description
Message	xsd:string	Text received in SMS
SenderAddress	xsd:anyURI	It indicates address sending the SMS
SmsServiceActivation Number	xsd:anyURI	Number associated with the invoked Message service, i.e. the destination address used to send the message.

8 Web Service Interface Definition

8.1 Interface : SendSms

8.1.1 Operation : SendSms

The invocation of **sendSms** requests to send an SMS, specified by the String **Message** to the specified address (or address set), specified by **Addresses**. Optionally the application can also indicate the sender name (**SenderName**), i.e. the string that is displayed on the user's terminal as the originator of the message, and the charging information. By invoking this operation the application requires to receive the notification of the status of the SMS delivery. In order to receive this information the application has to explicitly invoke the **getSmsDeliveryStatus**. The **RequestIdentifier**, returned by the invocation, can be used to identify the SMS delivery request.

Addresses may include group URIs as defined in the Address List Management specification. If groups are not supported, a PolicyException (POL0006) will be returned to the application.

For GSM systems, if **Message** contains characters not in the GSM 7-bit character set, the SMS is sent as a Unicode SMS.

If **Message** is longer than the maximum supported length (e.g. for GSM, 160 GSM 7-bit characters or 70 Unicode characters), the message will be sent as several concatenated short messages.

8.1.1.1 Input message : SendSmsRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	Addresses to which the SMS will be sent

SenderName	xsd:string	If present, it indicates the SMS sender name, i.e. the string that is displayed on the user's terminal as the originator of the message.
Charging	common:ChargingInformation	Charge to apply to this message (optional)
Message	xsd:string	Text to be sent in SMS

8.1.1.2 Output message : SendSmsResponse

Part Name	Part Type	Description
RequestIdentifier	xsd:string	It identifies a specific SMS delivery request

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0004 – No valid addresses
- SVC0006 – Invalid group
- SVC0280 – Message too long

PolicyException from [6]

- POL0001 – Policy error
- POL0006 – Groups not allowed
- POL0007 – Nested groups not allowed
- POL0008 – Charging not allowed

8.1.2 Operation : SendSmsLogo

The invocation of **sendSmsLogo** requests to send an SMS logo, specified by the byte array **image** to the specified address (or address set), specified by **destinationAddressSet**. Optionally the application can also indicate the sender name (**senderName**), i.e. the string that is displayed on the user's terminal as the originator of the message, and the charging information (**charging**). By invoking this operation the application requires to receive the notification of the status of the SMS delivery. In order to receive this information the application has to explicitly invoke the **getSmsDeliveryStatus**. The **requestIdentifier**, returned by the invocation, can be used to identify the SMS delivery request.

Addresses may include group URIs as defined in the Address List Management specification. If groups are not supported, a PolicyException (POL0006) will be returned to the application.

8.1.2.1 Input message : SendSmsLogoRequest

Part Name	Part Type	Descriptin
Addresses	xsd:anyURI [0..unbounded]	Addresses to which the SMS logo will be sent
SenderName	xsd:string	SMS sender name, i.e. the string that is displayed on the user's terminal as the originator of the message. (optional)

Charging	common:ChargingInformation	Charge to apply to this message. (optional)
Image	xsd:base64Binary	The image in jpeg, gif or png format. The image will be scaled to the proper format.
SmsFormat	SmsFormat	Possible values are: 'Ems' or 'SmartMessaging'.

8.1.2.2 Output message : SendSmsLogoResponse

Part Name	Part Type	Description
requestIdentifier	String	It identifies a specific SMS delivery request

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0004 – No valid addresses
- SVC0006 – Invalid group
- SVC0281 – Unrecognized data format

PolicyException from [6]

- POL0001 – Policy error
- POL0006 – Groups not allowed
- POL0007 – Nested groups not allowed
- POL0008 – Charging not allowed

8.1.3 Operation : SendSmsRingtone

The invocation of **sendSmsRingtone** requests to send an SMS ringtone, specified by the String **ringtone** (in RTX format) to the specified addresses, specified by **Addresses**. Optionally the application can also indicate the sender name (**senderName**) i.e. the string that is displayed on the user's terminal as the originator of the message, and the charging information (**charging**). By invoking this operation the application requires to receive the notification of the status of the SMS delivery. In order to receive this information the application has to explicitly invoke the **getSmsDeliveryStatus**. The **requestIdentifier**, returned by the invocation, can be used to identify the SMS delivery request.

Addresses may include group URIs as defined in the Address List Management specification. If groups are not supported, a PolicyException (POL0006) will be returned to the application.

Depending on the length of the ringtone, it may be sent as several concatenated short messages.

NOTE on the RTX Ringtone Specification : An RTX file is a text file, containing the ringtone name, a control subclause and a subclause containing a comma separated sequence of ring tone commands.

8.1.3.1 Input message : SendSmsRingtoneRequest

Part Name	Part Type	Descriptin
Addresses	xsd:anyURI [0..unbounded]	Addresses to which the SMS logo will be sent

SenderName	xsd:string	SMS sender name, i.e. the string that is displayed on the user's terminal as the originator of the message. (optional)
Charging	common:ChargingInformation	Charge to apply to this message. (optional)
Ringtone	xsd:string	The ringtone in RTX format (see Note above). (http://www.logomanager.co.uk/help/Edit/RTX.html)
SmsFormat	SmsFormat	Possible values are: 'Ems' or 'SmartMessaging'.

8.1.3.2 Output message : SendSmsRingtoneResponse

Part Name	Part Type	Description
RequestIdentifier	xsd:string	It identifies a specific SMS delivery request

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0004 – No valid addresses
- SVC0006 – Invalid group
- SVC0281 – Unrecognized data format

PolicyException from [6]

- POL0001 – Policy error
- POL0006 – Groups not allowed
- POL0007 – Nested groups not allowed
- POL0008 – Charging not allowed

8.1.4 Operation : GetSmsDeliveryStatus

The invocation of **getSmsDeliveryStatus** requests the status of a previous SMS delivery request identified by **requestIdentifier**. The information on the status is returned in **deliveryStatus**, which is an array of status related to the request identified by **requestIdentifier**. The status is identified by a couplet indicating a user address and the associated delivery status. This method can be invoked multiple times by the application even if the status has reached a final value. However, after the status has reached a final value, status information will be available only for a limited period of time that should be specified in an off-line configuration step. The following four different SMS delivery status have been identified:

- 'Delivered': in case of concatenated messages, only when all the SMS-parts have been successfully delivered.
- 'DeliveryUncertain': e.g. because it was handed off to another network.
- 'DeliveryImpossible': unsuccessful delivery; the message could not be delivered before it expired.
- 'MessageWaiting': the message is still queued for delivery.

8.1.4.1 Input message : GetSmsDeliveryStatusRequest

Part Name	Part Type	Description

RequestIdentifier	xsd:string	It identifies a specific SMS delivery request

8.1.4.2 Output message : GetSmsDeliveryStatusResponse

Part Name	Part Type	Description
DeliveryStatus	DeliveryInformation [0..unbounded]	It lists the variations on the delivery status of the SMS

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.2 Interface : SmsNotification

8.2.1 Operation : NotifySmsReception

The **notifySmsReception** method must be implemented by a Web Service at the ***application side***. It will be invoked by the Parlay X server to notify the application of the reception of an SMS. The notification will occur if and only if the SMS received fulfils the criteria specified in an off-line provisioning step, identified by the **registrationIdentifier**. The criteria must at least include an **smsServiceActivationNumber**, i.e. the SMS destination address that can be "monitored" by the application. The parameter **senderAddress** contains the address of the sender. The application can apply the appropriate service logic to process the SMS.

8.2.1.1 Input message : NotifySmsReceptionRequest

Part Name	Part Type	Description
RegistrationIdentifier	xsd:string	Identifies the off-line provisioning step that enables the application to receive notification of SMS reception according to specified criteria.
Message	SmsMessage	Message received

8.2.1.2 Output message : NotifySmsReceptionResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

None.

8.3 Interface : ReceiveSms

8.3.1 Operation : GetReceivedSms

The invocation of **getReceivedSms** retrieves all the SMS messages received that fulfil the criteria identified by **registrationIdentifier**. The method returns only the list of SMS messages received since the previous invocation of the same method, i.e. each time the method is executed the messages returned are removed from the server. Moreover, each

SMS message will be automatically removed from the server after a maximum time interval specified in an off-line configuration step.

The received SMS messages are returned in **receivedSms**. An SMS message is identified by a structure indicating the sender of the SMS message and the content.

8.3.1.1 Input message : GetReceivedSmsRequest

Part Name	Part Type	Description
RegistrationIdentifier	xsd:string	Identifies the off-line provisioning step that enables the application to receive notification of SMS reception according to specified criteria.

8.3.1.2 Output message : GetReceivedSmsResponse

Part Name	Part Type	Description
ReceivedSms	SmsMessage [0..unbounded]	It lists the received SMS since last invocation.

8.3.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

9 Fault Definitions

9.1 ServiceException

9.1.1 SVC0280: Message too long

Message Id	SVC0280
Text	Message too long. Maximum length is %1 characters.
Variables	%1 Number of characters allowed in a message

9.1.2 SVC0281: Unrecognized data format

Message Id	SVC0281
Text	Data format not recognized for message part %1.
Variables	%1 Message part with the unrecognized data

10 Service Policies

Service policies for this service.

Name	Type	Description
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions
ChargingSupported	xsd:boolean	Is charging supported for send operations

Annex A (normative): WSDL for Short Messaging

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-04-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-5 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 5: Multimedia Messaging
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 5
2 References 5
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 6
4 Detailed Service Description 6
5 Namespaces 7
6 Sequence Diagrams 7
6.1 Send Picture 7
7 XML Schema Data Type Definition 8
7.1 DeliveryStatus Enumeration 8
7.2 MessagePriority Enumeration 9
7.3 DeliveryInformation Structure 9
7.4 MessageReference Structure 9
7.5 MessageURI Structure 9
8 Web Service Interface Definition 10
8.1 Interface : SendMessage 10
8.1.1 Operation : SendMessage 10
8.1.1.1 Input message : SendMessageRequest 10
8.1.1.2 Output message : SendMessageResponse 10
8.1.1.3 Referenced Faults 10
8.1.2 Operation : GetMessageDeliveryStatus 11
8.1.2.1 Input message : GetMessageDeliveryStatusRequest 11
8.1.2.2 Output message : GetMessageDeliveryStatusResponse 11
8.1.2.3 Referenced Faults 11
8.2 Interface : ReceiveMessage 11
8.2.1 Operation : GetReceivedMessages 11
8.2.1.1 Input message : GetReceivedMessagesRequest 12
8.2.1.2 Output message : GetReceivedMessagesResponse 12
8.2.1.3 Referenced Faults 12
8.2.2 Operation : GetMessageURIs 12
8.2.2.1 Input message : GetMessageURIsRequest 12
8.2.2.2 Output message : GetMessageURIsResponse 12
8.2.2.3 Referenced Faults 12
8.2.3 Operation : GetMessage 13
8.2.3.1 Input message : GetMessageRequest 13
8.2.3.2 Output message : GetMessageResponse 13
8.2.3.3 Referenced Faults 13
8.3 Interface : MessageNotification 13
8.3.1 Operation : NotifyMessageReception 13
8.3.1.1 Input message : NotifyMessageReceptionRequest 13
8.3.1.2 Output message : NotifyMessageReceptionResponse 13
8.3.1.3 Referenced Faults 14
9 Fault Definitions 14
10 Service Policies 14
Annex A (normative): WSDL for Multimedia Messaging 15
Annex B (informative): Change history 16

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 5 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	**Multimedia Messaging**
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 5 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Multimedia Messaging Web Service aspects of the interface. All aspects of the Multimedia Messaging Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

[7] SOAP Messages with Attachments, available at http://www.w3.org/TR/SOAP-attachments

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] and the following apply.

EMS	Enhanced Messaging Service
IM	Instant Messaging
MMS	Multimedia Messaging Service
MMS-C	Multimedia Messaging Service Center
SMS	Short Messaging Service

4 Detailed Service Description

Currently, in order to programmatically receive and send Multimedia Messages, it is necessary to write applications using specific protocols to access MMS functions provided by network elements (e.g., MMS-C). This approach requires application developers to have a high degree of network expertise.

This contribution defines a Multimedia Messaging Web Service that can map to SMS, EMS, MMS, IM, E-mail etc.

The choice is between defining one set of interfaces per messaging network or a single set common to all networks; e.g. we could define sendMMS, sendEMS, sendSMS, … or just use sendMessage. Although the more specific the API the easier it is to use, there are advantages to a single set of network-neutral APIs. These advantages include:

- improved service portability
- lower complexity, by providing support for generic user terminal capabilities only.

For this version of the Parlay X specification, we provide sets of interfaces for two messaging web services: Short Messaging (part 7) and Multimedia Messaging (this part), which provides generic messaging features (including SMS).

For sending a message to the network (see Send Message), the application invokes a message to send it and must subsequently become active again to poll for delivery status. There is an alternative to this polling mechanism, i.e. an asynchronous notification mechanism implemented with an application-side web service. However it was decided not to provide a notification mechanism in the first release, to make the interface as simple as possible, even though the polling mechanism is not as network efficient as the notification mechanism.

For receiving a message from the network, the application may use either polling (see Receive Message) or notification (see Message Notification) mechanisms. The notification mechanism is more common: network-initiated messages are sent to autonomous application-side web services. Both mechanisms are supported, but the provisioning of the notification-related criteria is not specified.

Figure 1 shows an example scenario using sendMessage and getMessageDeliveryStatus to send data to subscribers and to determine if the data has been received by the subscriber. The application invokes a web service to retrieve a stock quote (1) & (2) and sends the current quote - sendMessage - using the Parlay X Interface (3) of the Multimedia Messaging Web Service. After invocation, the Multimedia Message Web Service sends the message to an MMS-C using the MM7 interface (4) for onward transmission (5) to the subscriber over the Mobile network

Later, when the next quote is ready, the application checks to see - getMessageDeliveryStatus - if the previous quote has been successfully delivered to the subscriber. If not, it may for instance perform an action (not shown) to provide a credit for the previous message transmission. This way, the subscriber is only charged for a stock quote if it is delivered on time.

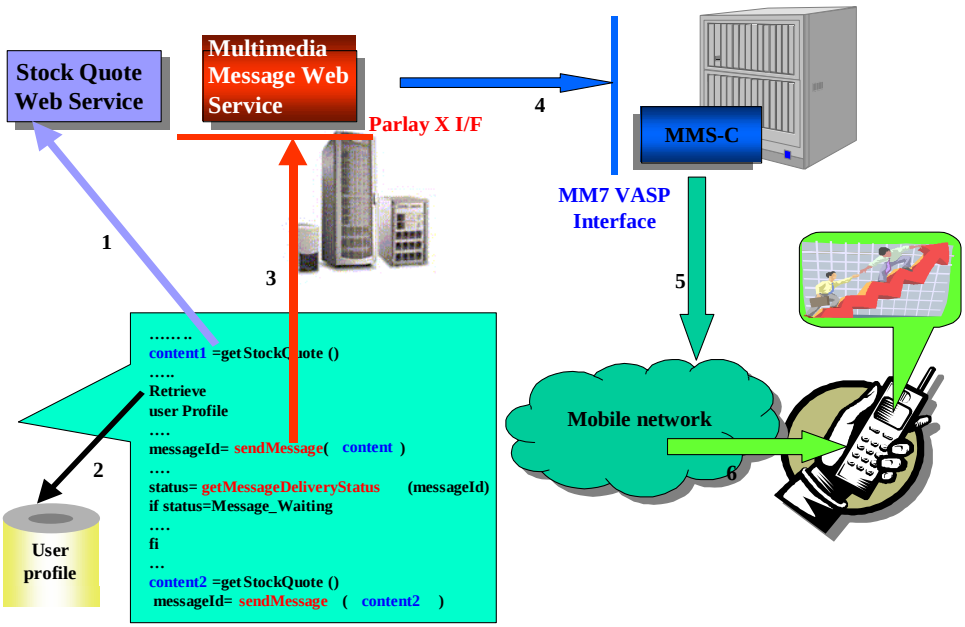


Figure 1: Multimedia Messaging Scenario

5 Namespaces

The SendMessage interface uses the namespace

www.csapi.org/wsdl/parlayx/multimedia_messaging/send/v2_0

The ReceiveMessage interface uses the namespace

www.csapi.org/wsdl/parlayx/multimedia_messaging/receive/v2_0

The MessageNotification interface uses the namespace

www.csapi.org/wsdl/parlayx/multimedia_messaging/notification/v2_0

The data types are defined in the namespace

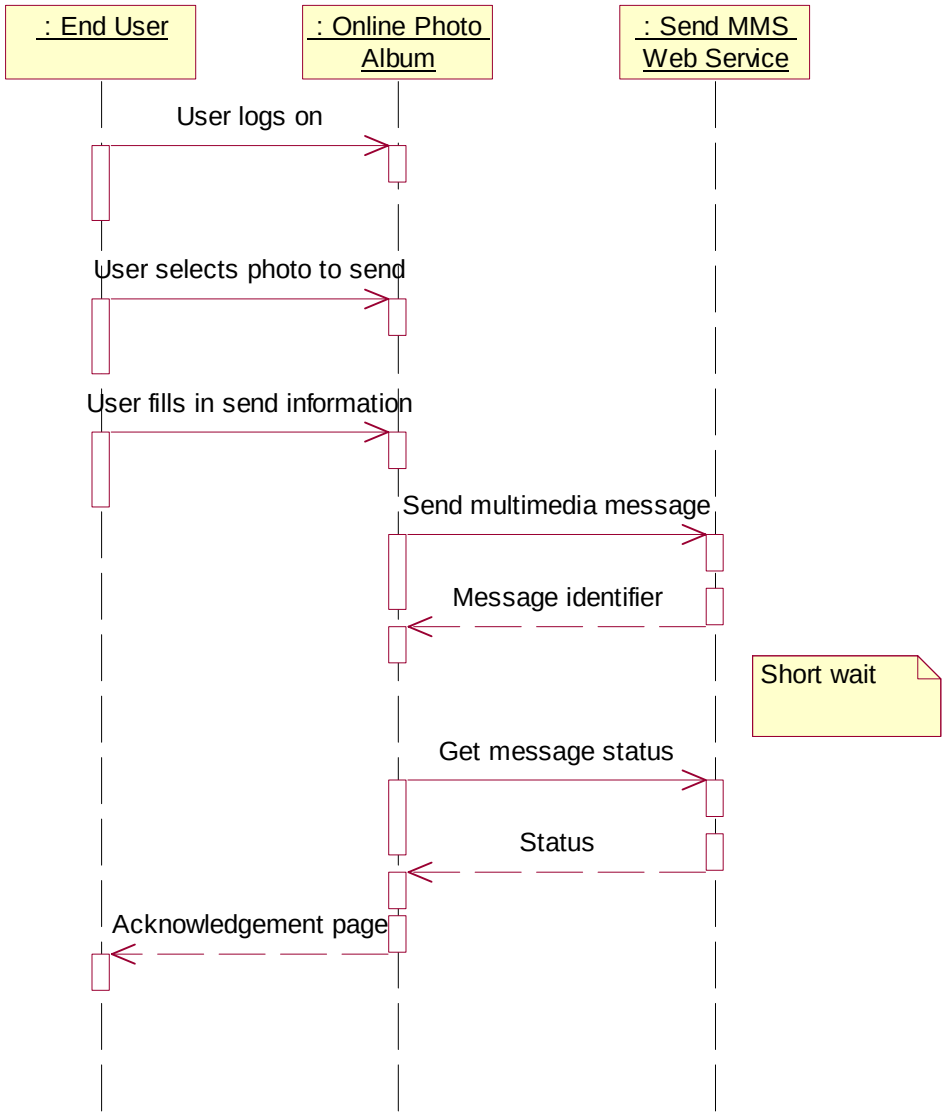
www.csapi.org/schema/parlayx/multimedia_messaging/v2_0

The 'xsd' namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name 'xsd' is not semantically significant.

6 Sequence Diagrams

6.1 Send Picture

With the advent of picture capable phones, the exchange of photos to mobile phones is becoming more common place. This sequence diagram shows an application where a person can send a picture from an online photo album to a mobile phone.



7 XML Schema Data Type Definition

7.1 DeliveryStatus Enumeration

List of delivery status values.

Enumeration	Description
Delivered	Successful delivery
DeliveryUncertain	Delivery status unknown: e.g. because it was handed off to another network.
DeliveryImpossible	Unsuccessful delivery; the message could not be delivered before it expired.
MessageWaiting	The message is still queued for delivery. This is a temporary state, pending transition to one of the preceding states.

7.2 MessagePriority Enumeration

List of delivery priority values.

Enumeration	Description
Default	Default message priority
Low	Low message priority
Normal	Normal message priority
High	High message priority

7.3 DeliveryInformation Structure

Delivery status information.

Element Name	Element Type	Description
address	xsd:anyURI	Address associated with the delivery status. The address field is coded as a URI.
deliveryStatus	DeliveryStatus	Parameter indicating the delivery status.

7.4 MessageReference Structure

Message information.

Element Name	Element Type	Description
messageIdentifier	xsd:string	OPTIONAL: If present, contains a reference to a message stored in the Parlay X gateway. If the message is pure text, this parameter is not present.
messageService ActivationNumber	xsd:string	Number associated with the invoked Message service, i.e. the destination address used by the terminal to send the message.
senderAddress	xsd:anyURI	Indicates message sender address
subject	xsd:string	OPTIONAL: If present, indicates the subject of the received message. This parameter will not be used for SMS services.
priority	MessagePriority	The priority of the message: default is Normal
message	xsd:string	OPTIONAL: If present, then the **messageIdentifier** is not present and this parameter contains the whole message. The type of the message is always pure ASCII text in this case. The message will not be stored in the Parlay X gateway.

7.5 MessageURI Structure

Message location information.

Element Name	Element Type	Description
bodyText	xsd:string	Contains the message body if it is encoded as ASCII text.
fileReferences	xsd:anyURI [0..unbounded]	This is an array of URI references to all the attachments in the Multimedia message. These are URIs to different files, e.g. GIF pictures or pure text files.

8 Web Service Interface Definition

8.1 Interface : SendMessage

Operations to send messages and check status on sent messages.

8.1.1 Operation : SendMessage

Request to send a Message to a set of destination addresses, returning a **requestIdentifier** to identify the message. The **requestIdentifier** can subsequently be used by the application to poll for the message status, i.e. using **getMessageDeliveryStatus** to see if the message has been delivered or not. The content is sent as a Attachment as specified in SOAP Messages with Attachments [7].

Addresses may include group URIs as defined in the Address List Management specification. If groups are not supported, a PolicyException (POL0006) will be returned to the application.

8.1.1.1 Input message : SendMessageRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	Destination addresses for the Message.
SenderAddress	xsd:string	Message sender address. This parameter is not allowed for all 3rd party providers. Parlay X server needs to handle this according to a SLA for the specific application and its use can therefore result in a PolicyException. (optional)
Subject	xsd:string	Message subject. If mapped to SMS this parameter will be used as the senderAddress, even if a separate senderAddress is provided. (optional)
Priority	MessagePriority	Priority of the message. If not present, the network will assign a priority based on an operator policy. (optional)
Charging	Common:Charging Information	Charging to apply to this message. (optional)

NOTE: The input message may also contain attachments, with appropriate content as defined by SOAP Messages withAttachments [7].

8.1.1.2 Output message : SendMessageResponse

Part Name	Part Type	Description
RequestIdentifier	xsd:string	It is a correlation identifier that is used in a **getMessageDeliveryStatus** message invocation, i.e. to poll for the delivery status of all of the sent Messages.

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0004 – No valid addresses

- SVC0006 – Invalid group

PolicyException from [6]

- POL0001 – Policy error
- POL0006 – Groups not allowed
- POL0007 – Nested groups not allowed
- POL0008 – Charging not supported

8.1.2 Operation : GetMessageDeliveryStatus

This is a poll method used by the application to retrieve delivery status for each message sent as a result of a previous **sendMessage** message invocation. The **requestIdentifier** parameter identifies this previous message invocation.

8.1.2.1 Input message : GetMessageDeliveryStatusRequest

Part Name	Part Type	Description
RequestIdentifier	xsd:string	Identifier related to the delivery status request.

8.1.2.2 Output message : GetMessageDeliveryStatusResponse

Part Name	Part Type	Description
DeliveryStatus	DeliveryInformation [0..unbounded]	It is an array of status of the messages that were previously sent. Each array element represents a sent message: i.e. its destination address and its delivery status.

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.2 Interface : ReceiveMessage

Operations to retrieve messages that have been received.

8.2.1 Operation : GetReceivedMessages

This method enables the application to poll for new messages associated with a specific **registrationIdentifier**. If the **registrationIdentifier** is not specified, the Parlay X server will return references to all messages sent to the application. The process of binding different **registrationIdentifier** parameters to applications is an off-line process. The Parlay X gateway shall not allow an application to poll for messages using **registrationIdentifier** parameters that are not associated with the application. The priority parameter may be used by the application to retrieve references to higher priority messages, e.g. if Normal is chosen only references to high priority and normal priority messages are returned. If the priority parameter is omitted all message references are returned.

8.2.1.1 Input message : GetReceivedMessagesRequest

Part Name	Part Type	Description
RegistrationIdentifier	xsd:string	Identifies the off-line provisioning step that enables the application to receive notification of Message reception according to specified criteria.
Priority	MessagePriority	OPTIONAL. The priority of the messages to poll from the Parlay X gateway. All messages of the specified priority and higher will be retrieved. If not specified, all messages shall be returned, i.e. the same as specifying Low.

8.2.1.2 Output message : GetReceivedMessagesResponse

Part Name	Part Type	Description
Messages	MessageReference [0..unbounded]	It contains an array of messages received according to the specified filter of **registrationIdentifier** and **priority**.

8.2.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.2.2 Operation : GetMessageURIs

This method will read the different parts of the message, create local files in the Parlay Gateway and return URI references to them. The application can then simply read each file or just have them presented as links to the end-user. The URIs to the files will be active for an agreed time.

8.2.2.1 Input message : GetMessageURIsRequest

Part Name	Part Type	Description
MessageRefIdentifier	xsd:string	The identity of the message to retrieve.

8.2.2.2 Output message : GetMessageURIsResponse

Part Name	Part Type	Description
Message	MessageURI	It contains the complete message, i.e. the textual part of the message, if such exists, and a list of file references for the message attachments, if any.

8.2.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.2.3 Operation : GetMessage

This method will read the whole message. The data is returned as an attachment, as defined in SOAP Messages with Attachments [7], in the return message.

8.2.3.1 Input message : GetMessageRequest

Part Name	Part Type	Description
MessageRefIdentifier	String	The identity of the message

8.2.3.2 Output message : GetMessageResponse

Part Name	Part Type	Description
None		

8.2.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.3 Interface : MessageNotification

8.3.1 Operation : NotifyMessageReception

This method will have to be implemented by a web service on the client application side. The registration of the URI for this application web service is done off-line. This means that there is a registration mechanism in the Parlay X Gateway that binds different **registrationIdentifier** parameters to applications and their web service URIs.

A client application is notified that a new Message, sent to a specific Service Activation Number, has been received. Using the **registrationIdentifier**, the client application can apply appropriate service logic with specific behaviour.

8.3.1.1 Input message : NotifyMessageReceptionRequest

Part Name	Part Type	Description
RegistrationIdentifier	xsd:string	A handle connected to the off-line registration of the notifications. This distinguishes registrations that point to the same application web service.
Message	MessageReference	This parameter contains all the information associated with the received message.

8.3.1.2 Output message : NotifyMessageReceptionResponse

Part Name	Part Type	Description

None		

8.3.1.3 Referenced Faults

None.

9 Fault Definitions

No new faults are defined by this service.

10 Service Policies

Service policies for this service.

Name	Type	Description
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions
ChargingSupported	xsd:boolean	Charging supported for send message operation

Annex A (normative): WSDL for Multimedia Messaging

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-05-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-6 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services; Part 6: Payment (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 7
6 Sequence Diagrams 8
6.1 Charge for Content 8
7 XML Schema Data Type Definition 8
7.1 Property Structure 8
8 Web Service Interface Definition 9
8.1 Interface : AmountCharging 9
8.1.1 Operation : ChargeAmount 9
8.1.1.1 Input message : ChargeAmountRequest 9
8.1.1.2 Output message : ChargeAmountResponse 9
8.1.1.3 Referenced Faults 9
8.1.2 Operation : RefundAmount 9
8.1.2.1 Input message : RefundAmountRequest 9
8.1.2.2 Output message : RefundAmountResponse 10
8.1.2.3 Referenced Faults 10
8.2 Interface : VolumeCharging 10
8.2.1 Operation : ChargeVolume 10
8.2.1.1 Input message : ChargeVolumeRequest 10
8.2.1.2 Output message : ChargeVolumeResponse 10
8.2.1.3 Referenced Faults 11
8.2.2 Operation : GetAmount 11
8.2.2.1 Input message : GetAmountRequest 11
8.2.2.2 Output message : GetAmountResponse 11
8.2.2.3 Referenced Faults 11
8.2.3 Operation : RefundVolume 11
8.2.3.1 Input message : RefundVolumeRequest 12
8.2.3.2 Output message : RefundVolumeResponse 12
8.2.3.3 Referenced Faults 12
8.3 Interface : ReserveAmountCharging 12
8.3.1 Operation : ReserveAmount 12
8.3.1.1 Input message : ReserveAmountRequest 12
8.3.1.2 Output message : ReserveAmountResponse 13
8.3.1.3 Referenced Faults 13
8.3.2 Operation : ReserveAdditionalAmount 13
8.3.2.1 Input message : ReserveAdditionalAmountRequest 13
8.3.2.2 Output message : ReserveAdditionalAmountResponse 13
8.3.2.3 Referenced Faults 13
8.3.3 Operation : ChargeReservation 14
8.3.3.1 Input message : ChargeReservationRequest 14
8.3.3.2 Output message : ChargeReservationResponse 14
8.3.3.3 Referenced Faults 14
8.3.4 Operation : ReleaseReservation 14
8.3.4.1 Input message : ReleaseReservationRequest 14
8.3.4.2 Output message : ReleaseReservationResponse 14
8.3.4.3 Referenced Faults 15
8.4 Interface : ReserveVolumeCharging 15
8.4.1 Operation : GetAmount 15

8.4.1.1 Input message : GetAmountRequest 15
8.4.1.2 Output message : GetAmountResponse 15
8.4.1.3 Referenced Faults 15
8.4.2 Operation : ReserveVolume 16
8.4.2.1 Input message : ReserveVolumeRequest 16
8.4.2.2 Output message : ReserveVolumeResponse 16
8.4.2.3 Referenced Faults 16
8.4.3 Operation : ReserveAdditionalVolume 16
8.4.3.1 Input message : ReserveAdditionalVolumeRequest 16
8.4.3.2 Output message : ReserveAdditionalVolumeResponse 17
8.4.3.3 Referenced Faults 17
8.4.4 Operation : ChargeReservation 17
8.4.4.1 Input message : ChargeReservationRequest 17
8.4.4.2 Output message : ChargeReservationResponse 17
8.4.4.3 Referenced Faults 17
8.4.5 Operation : ReleaseReservation 18
8.4.5.1 Input message : ReleaseReservationRequest 18
8.4.5.2 Output message : ReleaseReservationResponse 18
8.4.5.3 Referenced Faults 18
9 Fault Definitions 18
9.1 ServiceException 18
9.1.1 SVC0270: Charge failed 18
10 Service Policies 18
Annex A (normative): WSDL for Payment 19
Annex B (informative): Change history 20

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 6 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	**Payment**
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 6 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Payment Web Service aspects of the interface. All aspects of the Payment Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

A vast amount of content, both information and entertainment, will be made available to subscribers. To support a business model that enables operators to offer integrated billing, a payment API is crucial. Open and inter-operable "payment APIs" are the key to market growth and investment protection. The Payment Web Service supports payments for any content in an open, Web-like environment.

The Payment Web Service described in this document supports payment reservation, pre-paid payments, and post-paid payments. It supports charging of both volume and currency amounts, a conversion function and a settlement function in case of a financially resolved dispute.

Note that certain parameters are negotiated off line. For example the currency, volume type, default reservation enforcement time, as well as the taxation procedures and parameters.

An example of an application scenario could be a multimedia service. Assume a subscriber is interested in receiving a stream of, say, a soccer match. The subscriber selects a match and establishes a trusted relation with the provider. Again, the provider obtains the MSISDN and other information from the subscriber. The subscriber wants to know what the service will cost and the provider interacts with the operators rating engine (**getAmount**) taking into account the subscriber's subscription, time of day, etc. The value returned is a currency amount and is printed on the page that is displayed at the MS. The subscriber then decides to stream the match to his MS. Subsequently, the provider will reserve the appropriate amount with the operator (**reserveAmount**) to ensure that the subscriber can fulfil his payment obligations. The match starts and the provider periodically charges against the reservation (**chargeReservation**). The match ends in a draw and is extended with a 'sudden death' phase. The subscriber continues listening, so the existing reservation is enlarged (**reserveAdditionalAmount**). Suddenly, one of the teams scores a goal, so the match abruptly ends, leaving part of the reserved amount unused. The provider now releases the reservation (**releaseReservation**), and the remaining amount is available for future use by the subscriber.

Now we can extend this scenario by having the subscriber participate in a game of chance in which the provider refunds a percentage of the usage costs (**refundAmount**) based on the ranking of a particular team in this tournament. For example, the subscriber gambling on the team that wins the tournament receives a full refund, while for gambling on the team that finishes in second place, the refund is 50%, etc.

5 Namespaces

The AmountCharging interface uses the namespace

www.csapi.org/wsdl/parlayx/payment/amount_charging/v2_0

The VolumeCharging interface uses the namespace

www.csapi.org/wsdl/parlayx/payment/volume_charging/v2_0

The ReserveAmountCharging interface uses the namespace

www.csapi.org/wsdl/parlayx/payment/reserve_amount_charging/v2_0

The ReserveVolumeCharging interface uses the namespace

www.csapi.org/wsdl/parlayx/payment/reserve_volume_charging/v2_0

The data types are defined in the namespace

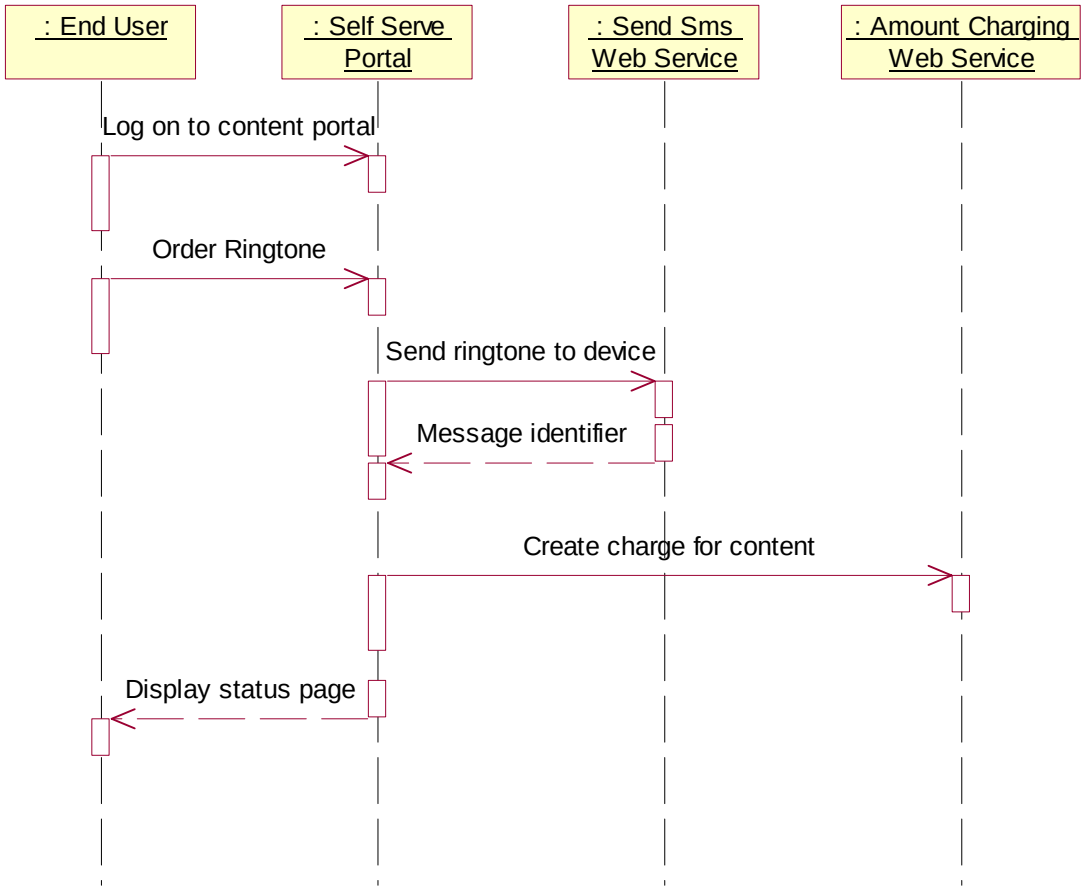
www.csapi.org/schema/parlayx/payment/v2_0

The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Charge for Content

Assume a subscriber is interested in downloading a ring tone to his device. The subscriber selects a ring tone and establishes a trusted relation with the ring tone provider. Essentially, the ring tone provider obtains the address (MSISDN) and other information from the subscriber. The ring tone may be downloaded to the device using SMS. As soon as the download succeeds, the provider of the ring tone will charge the subscriber (**chargeAmount)**.



7 XML Schema Data Type Definition

7.1 Property Structure

Property with a name and value.

Name	Type	Description
Name	xsd:string	Name of property
Value	xsd:string	Value of property

8 Web Service Interface Definition

8.1 Interface : AmountCharging

Charge operations by amount.

8.1.1 Operation : ChargeAmount

This message results in directly charging to the account indicated by the end user identifier. The charge is specified as a currency amount. The billing text field is used for textual information to appear on the bill. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.1.1.1 Input message : ChargeAmountRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be charged
Amount	xsd:decimal	The currency amount of the charge
billingText	xsd:string	Textual information to appear on the bill
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.1.1.2 Output message : ChargeAmountResponse

Part Name	Part Type	Description
None		

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.1.2 Operation : RefundAmount

This message results in directly applying a refund to the account indicated by the end user identifier. The refund is specified as a currency amount. The billing text field is used for textual information to appear on the bill. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.1.2.1 Input message : RefundAmountRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be refunded

Amount	xsd:decimal	The currency amount of the refunded
billingText	xsd:string	Textual information to appear on the bill
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.1.2.2 Output message : RefundAmountResponse

Part Name	Part Type	Description
None		

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.2 Interface : VolumeCharging

Charging operations by volume.

8.2.1 Operation : ChargeVolume

This message results in directly charging to the account indicated by the end user identifier. The charge is specified as a volume. The billing text field is used for textual information to appear on the bill. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.2.1.1 Input message : ChargeVolumeRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be charged
volume	xsd:long	The volume to be charged
billingText	xsd:string	Textual information to appear on the bill
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.2.1.2 Output message : ChargeVolumeResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.2.2 Operation : GetAmount

This message results in converting the given volume to a currency amount. The end user identifier is given to indicate the subscriber for whom this conversion calculation must be made. The message returns a currency amount if successful.

The following properties may be provided;

- unit, specifying the unit used for measuring volume (e.g. bytes)
- contract, number of a contract that may govern the use
- service, name of the service to be used (e.g. SendMultimediaMessage)
- operation, name of the operation to be used (e.g. SendMessage)

8.2.2.1 Input message : GetAmountRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be charged
volume	xsd:long	The volume to be converted
parameters	Property [0..unbounded]	Parameters to use to perform rating (“unit”, “contract”, “service”, “operation”)

8.2.2.2 Output message : GetAmountResponse

Part Name	Part Type	Description
Result	xsd:decimal	It is the currency amount resulting from the conversion process

8.2.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.2.3 Operation : RefundVolume

This message results in directly applying a refund to the account indicated by the end user identifier. The refund is specified as a volume. The billing text field is used for textual information to appear on the bill. The reference code is

used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.2.3.1 Input message : RefundVolumeRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be refunded
volume	xsd:long	The volume to be refunded
billingText	xsd:string	Textual information to appear on the bill
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.2.3.2 Output message : RefundVolumeResponse

Part Name	Part Type	Description
None		

8.2.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.3 Interface : ReserveAmountCharging

Operations to manage reservation charging by amount.

8.3.1 Operation : ReserveAmount

This message results in directly reserving an amount for an account indicated by the end user identifier. The reservation is specified as a currency amount. Note that reservations do not last forever; it is assumed the default reservation enforcement time is negotiated off-line. If the reservation times out, the remaining funds will be returned to the account from which this reservation was made. However, the remaining funds shall preferably be returned explicitly to the account using the **releaseReservation** message. The billing text field is used for textual information to appear on the bill. Subsequent textual information provided during this charging session will be appended to this textual information; one charging session to a reservation will result in only one entry on the bill. In case of success, a reservation id is returned for future reference; e.g. subsequent charging against the existing reservation using the **chargeReservation** message.

8.3.1.1 Input message : ReserveAmountRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account subject to the reservation
amount	xsd:decimal	The currency amount of the reservation

billingText	xsd:string	Textual information to appear on the bill

8.3.1.2 Output message : ReserveAmountResponse

Part Name	Part Type	Description
reservationIdentifier	xsd:string	It is an identifier for the newly created reservation

8.3.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.3.2 Operation : ReserveAdditionalAmount

This message results in the addition/reduction of a currency amount to/from an existing reservation indicated by the reservation id. The reservation is specified as a currency amount. Note that reservations do not last forever; it is assumed the default reservation enforcement time is negotiated off-line. Invoking this message will extend the reservation enforcement time for another off-line-negotiated period. The billing text field is used for appending textual information to appear on the bill. The textual information is appended to the initial textual information given by the **reserveAmount** message; one charging session to a reservation will result in only one entry on the bill. Reserved credit can be returned to the account through the **releaseReservation** message.

8.3.2.1 Input message : ReserveAdditionalAmountRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be amended
amount	xsd:decimal	The currency amount to be added to (or subtracted from) the reservation
billingText	xsd:string	Textual information to appear on the bill

8.3.2.2 Output message : ReserveAdditionalAmountResponse

Part Name	Part Type	Description
None.		

8.3.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.3.3 Operation : ChargeReservation

This message results in charging to a reservation indicated by the reservation id. Reservations, identified by reservation id, are established through invoking the **reserveAmount** message. The charge is specified as a currency amount. Optionally, the billing text field can be used for appending textual information to appear on the bill. The textual information is appended to the initial textual information given by the **reserveAmount** message; one charging session to a reservation will result in only one entry on the bill. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.3.3.1 Input message : ChargeReservationRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be charged
amount	xsd:decimal	The currency amount of the charge
billingText	xsd:string	Textual information to appear on the bill
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.3.3.2 Output message : ChargeReservationResponse

Part Name	Part Type	Description
None		

8.3.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.3.4 Operation : ReleaseReservation

Returns funds left in a reservation indicated by reservation id to the account from which this reservation was made. Reservations, identified by reservation id, are established by invoking the reserveAmount message.

8.3.4.1 Input message : ReleaseReservationRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be released

8.3.4.2 Output message : ReleaseReservationResponse

Part Name	Part Type	Description
None		

8.3.4.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.4 Interface : ReserveVolumeCharging

Operations to manage reservation charging by amount.

8.4.1 Operation : GetAmount

Returns the amount resulting from converting the given volume. The end user identifier is given to indicate the subscriber for whom this calculation must be made. The message returns a currency amount if successful.

The following properties may be provided;

- unit, specifying the unit used for measuring volume (e.g. bytes)
- contract, number of a contract that may govern the use
- service, name of the service to be used (e.g. SendMultimediaMessage)
- operation, name of the operation to be used (e.g. SendMessage)

8.4.1.1 Input message : GetAmountRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account to be charged
volume	xsd:long	The volume to be converted
parameters	Property [0..unbounded]	Parameters to use to perform rating (“unit”, “contract”, “service”, “operation”)

8.4.1.2 Output message : GetAmountResponse

Part Name	Part Type	Description
amount	xsd:decimal	It is the currency amount resulting from the conversion process

8.4.1.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.4.2 Operation : ReserveVolume

Reserves an amount of an account indicated by the end user identifier. The reservation is specified as a volume. Note that reservations do not last forever; it is assumed the default reservation enforcement time is negotiated off-line. If the reservation times out, the remaining volume will be returned to the account from which this reservation was made. However, the remaining volume should preferably be returned explicitly to the account using the **releaseReservation** message. The billing text field is used for textual information to appear on the bill. Subsequent textual information provided during this charging session will be appended to this textual information; one charging session to a reservation will result in only one entry on the bill. In case of success, a reservation identifier is returned for future reference; e.g. subsequent charging against the existing reservation using the **chargeReservation** message.

8.4.2.1 Input message : ReserveVolumeRequest

Part Name	Part Type	Description
endUserIdentifier	xsd:anyURI	The end user's account subject to the reservation
volume	xsd:long	The volume of the reservation
billingText	xsd:string	Textual information to appear on the bill

8.4.2.2 Output message : ReserveVolumeResponse

Part Name	Part Type	Description
reservationIdentifier	xsd:string	It is an identifier for the newly created reservation

8.4.2.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.4.3 Operation : ReserveAdditionalVolume

Adds/reduces a volume to an existing reservation indicated by the reservation id. The reservation is specified as a volume. Note that reservations do not last forever; it is assumed the default reservation enforcement time is negotiated off-line. Invoking this message will extend the reservation enforcement time for another off-line-negotiated period. The billing text field is used for appending textual information to appear on the bill. The textual information is appended to the initial textual information given by the **reserveVolume** message; one charging session to a reservation will result in only one entry on the bill. A reserved credit can be returned to the account through the **releaseReservation** message.

8.4.3.1 Input message : ReserveAdditionalVolumeRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be amended
volume	xsd:long	The volume to be added to (or subtracted from) the reservation
billingText	xsd:string	Textual information to appear on the bill

8.4.3.2 Output message : ReserveAdditionalVolumeResponse

Part Name	Part Type	Description
None		

8.4.3.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

8.4.4 Operation : ChargeReservation

This message results in charging to a reservation indicated by the reservation id.. Reservations, identified by reservation id., are established through invoking the **reserveVolume** message. The charge is specified as a volume. Optionally, the billing text field can be used for appending textual information to appear on the bill. The textual information is appended to the initial textual information given by the **reserveVolume** message; one charging session to a reservation will result in only one entry on the bill. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application.

8.4.4.1 Input message : ChargeReservationRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be charged
volume	xsd:long	The currency amount of the charge
billingText	xsd:string	Textual information to appear on the bill (optional)
referenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes

8.4.4.2 Output message : ChargeReservationResponse

Part Name	Part Type	Description
None		

8.4.4.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value
- SVC0270 – Charge failed

PolicyException from [6]

- POL0001 – Policy error

8.4.5 Operation : ReleaseReservation

Returns funds left in a reservation indicated by reservation id. to the account from which this reservation was made. Reservations, identified by reservation id., are established through invoking the **reserveVolume** message.

8.4.5.1 Input message : ReleaseReservationRequest

Part Name	Part Type	Description
reservationIdentifier	xsd:string	An identifier for the reservation to be released

8.4.5.2 Output message : ReleaseReservationResponse

Part Name	Part Type	Description
None		

8.4.5.3 Referenced Faults

ServiceException from [6]

- SVC0001 – Service error
- SVC0002 – Invalid input value

PolicyException from [6]

- POL0001 – Policy error

9 Fault Definitions

9.1 ServiceException

9.1.1 SVC0270: Charge failed

Message Id	SVC0270
Text	Charging operation failed, the charge was not applied.
Variables	None

10 Service Policies

Name	Type	Description
Currency	xsd:string	Currency used by service (per ISO 4217)

Annex A (normative): WSDL for Payment

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-06-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-7 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services; Part 7: Account Management (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 5
2 References 5
3 Definitions and Abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 6
4 Detailed Service Description 6
5 Namespaces 6
6 Sequence Diagrams 6
6.1 Prepaid Account Recharge Using a Voucher 6
6.2 Prepaid Account Recharge Using Direct Payment 7
7 XML Schema Data Type Definition 8
7.1 DatedTransaction Structure 8
8 Web Service Interface Definition 9
8.1 Interface : AccountManagement 9
8.1.1 Operation : GetBalance 9
8.1.1.1 Input message : GetBalanceRequest 9
8.1.1.2 Output message : GetBalanceResponse 9
8.1.1.3 Referenced Faults 9
8.1.2 Operation : GetCreditExpiryDate 9
8.1.2.1 Input message : GetCreditExpiryDateRequest 9
8.1.2.2 Output message : GetCreditExpiryDateResponse 10
8.1.2.3 Referenced Faults 10
8.1.3 Operation : BalanceUpdate 10
8.1.3.1 Input message : BalanceUpdateRequest 10
8.1.3.2 Output message : BalanceUpdateResponse 10
8.1.3.3 Referenced Faults 10
8.1.4 Operation : VoucherUpdate 11
8.1.4.1 Input message : VoucherUpdateRequest 11
8.1.4.2 Output message : VoucherUpdateResponse 11
8.1.4.3 Referenced Faults 11
8.1.5 Operation : GetHistory 11
8.1.5.1 Input message : GetHistoryRequest 12
8.1.5.2 Output message : GetHistoryResponse 12
8.1.5.3 Referenced Faults 12
9 Fault Definitions 12
9.1 Fault : ServiceException 12
9.1.1 End user authentication failed. 12
9.1.2 Unknown Voucher 13
9.2 Fault : PolicyException 13
9.2.1 Vouchers not accepted 13
10 Service Policies 13
Annex A (normative): WSDL for Account Management 14
Annex B (informative): Change history 15

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 7 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	**Account Management**
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 7 of the Stage 3 Parlay X Web Service specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Account Management Web Service aspects of the interface. All aspects of the Account Management Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault defenitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

Pre-paid subscribers, whether they have subscribed to pre-paid telephony, SMS, or data service, have credits with their service providers; the consumption of services will lead to reduction of their credit, or the credit may expire. Therefore, from time to time, subscribers may have to recharge their accounts. This occurs through an application that interfaces with the subscriber either directly or indirectly. Examples of direct interaction are voice prompts and WAP/web pages, or even SMS. Typically, such multi-modal applications either request a currency amount and, e.g. credit card information, or a voucher number plus credentials. The voucher number and credentials are then validated and causes a pre-determined currency amount to be transferred.

The Parlay X Account Management API described in this document supports account querying, direct recharging and recharging through vouchers. As a side effect, it may prevent subscribers from having their account balance credits expire.

5 Namespaces

The Account Management interface uses the namespace

www.csapi.org/wsdl/parlayx/ account_management /v2_0

The data types are defined in the namespace

www.csapi.org/schema/parlayx/ account_management /v2_0

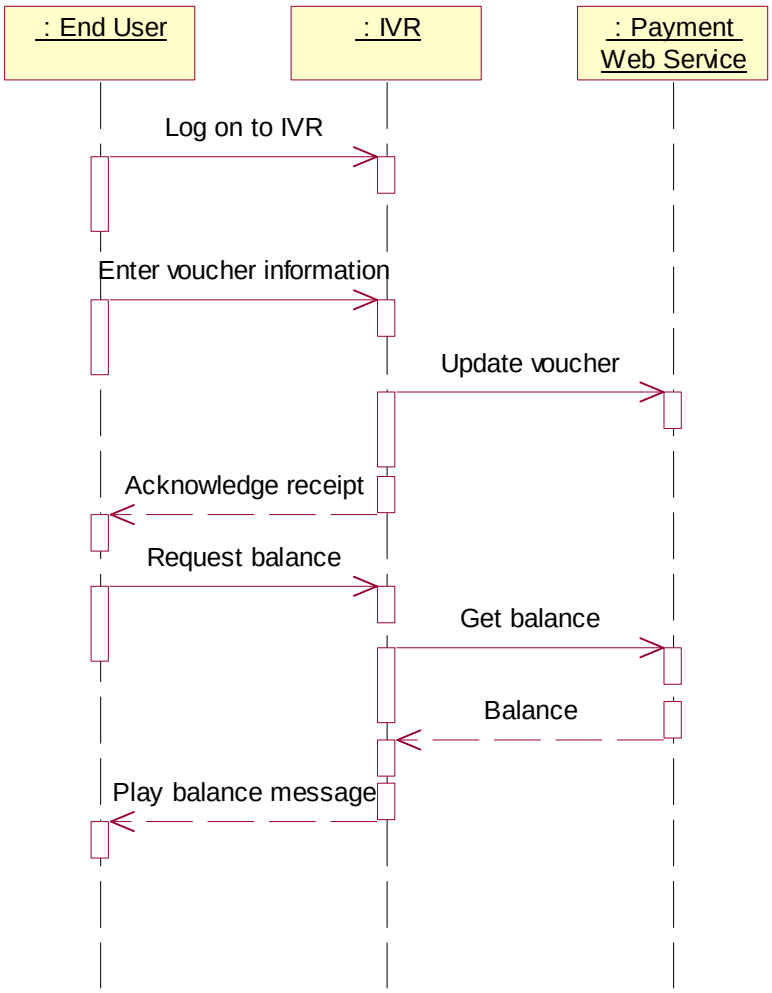
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

This subclause discusses three scenarios; one where a subscriber uses a voucher, one where the subscriber directly recharges after the payment is cleared, and one where the subscriber checks the recent transactions. Note, associated Account Management API messages are shown in 'bold' format: e.g. (**getBalance**).

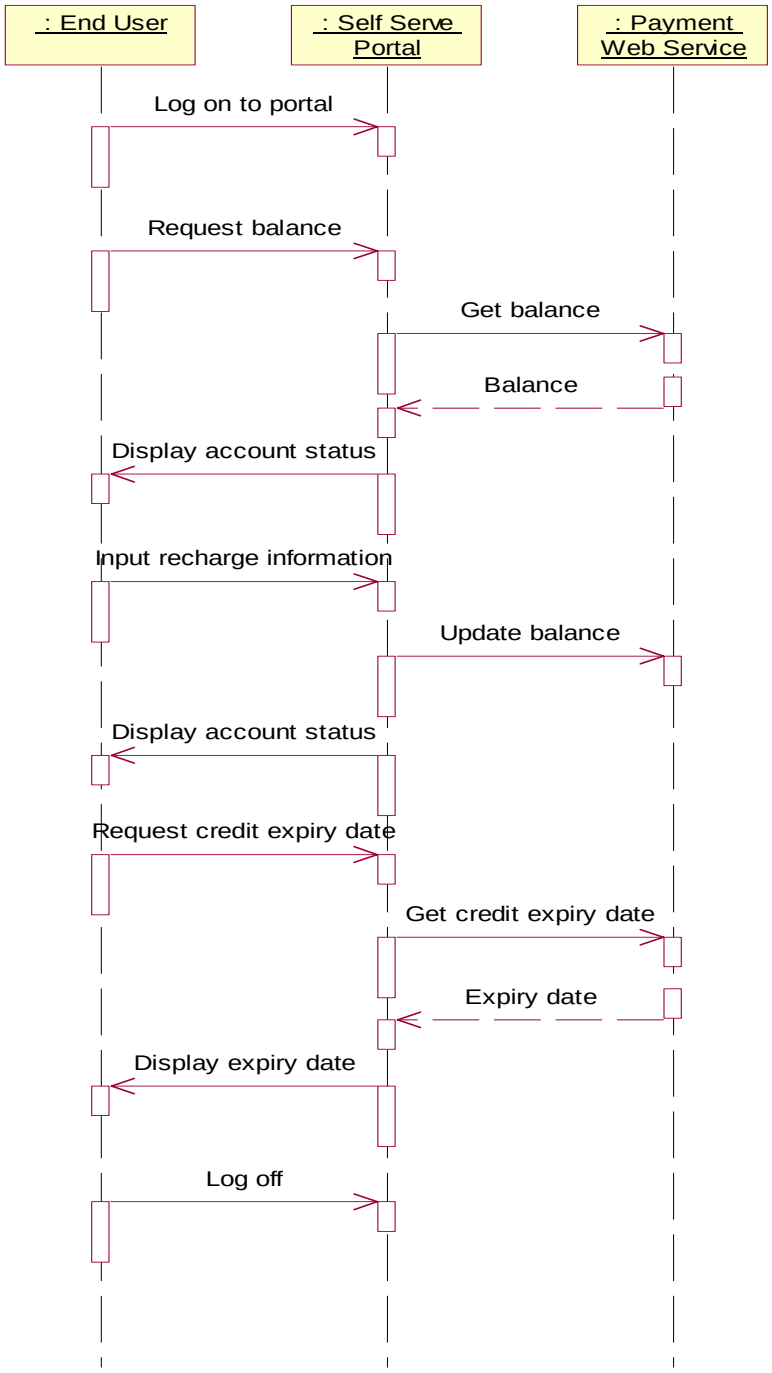
6.1 Prepaid Account Recharge Using a Voucher

The prepaid subscriber wishes to recharge their account with a voucher and query their account balance. The subscriber uses their mobile phone or other wireline phone to interact with an IVR system. In order to recharge their account, the subscriber must enter the voucher number, the MSISDN to be recharged , and PIN(s). The IVR system accesses an external voucher database to validate the voucher number. The subscriber's account balance is then increased with the value of the voucher (**voucherUpdate**). The subscriber queries their account balance (**getBalance**), before and/or after the recharge.



6.2 Prepaid Account Recharge Using Direct Payment

Directly recharging (i.e. without a voucher) works much along the same way. In this case, we assume the prepaid subscriber interacts with a web page. After providing the MSISDN, along with the PIN, the user can query the account balance (**getBalance**). For recharging, the subscriber must enter payment details, for example credit card information, from which the payment will be made. After clearing the payment details, the currency amount will be transferred and the subscriber's prepaid account balance expiration date will be reset (**balanceUpdate**). The subscriber also queries their account balance expiration date (**getCreditExpiryDate**), after the recharge.



7 XML Schema Data Type Definition

7.1 DatedTransaction Structure

This data structure represents a tranaction record.

Element Name	Element Type	Description
TransactionDate	xsd:dateTime	The date the transaction occurred.

TransactionDetails	xsd:string	The transaction details.

8 Web Service Interface Definition

8.1 Interface : AccountManagement

The Account Management interface provides access to account information for update and query operations.

8.1.1 Operation : GetBalance

This message results in getting account balance indicated by the end user identifier and associated end user PIN. The returned amount is specified as a currency amount.

8.1.1.1 Input message : GetBalanceRequest

Part Name	Part Type	Description
EndUserIdentifier	xsd:anyURI	This parameter identifies the end user's account.
EndUserPin	xsd:string	OPTIONAL: Contains the end user's credentials for authorizing access to the account

8.1.1.2 Output message : GetBalanceResponse

Part Name	Part Type	Description
Amount	xsd:decimal	It is the balance on the end user's account.

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0250: End user authentication failed

PolicyException from [6]

- POL0001 – Policy error

8.1.2 Operation : GetCreditExpiryDate

This message results in getting the expiration date of the credit indicated by the end user identifier and associated end user PIN. The returned date is the date the current balance will expire. Nil is returned if the balance does not expire.

8.1.2.1 Input message : GetCreditExpiryDateRequest

Part Name	Part Type	Description
EndUserIdentifier	xsd:anyURI	This parameter identifies the end user's account.
EndUserPin	xsd:string	OPTIONAL: Contains the end user's credentials for authorizing access to the account.

8.1.2.2 Output message : GetCreditExpiryDateResponse

Part Name	Part Type	Description
Date	xsd:dateTime	It is the date the current balance will expire. Nil is returned if the balance does not expire.

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0250: End user authentication failed

PolicyException from [6]

- POL0001 – Policy error

8.1.3 Operation : BalanceUpdate

This message results in directly recharging the account indicated by the end user identifier and optional associated end user PIN. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application. The charge is specified as a currency amount. The balance is requested to expire in the number of days indicated by the period parameter. The operator's policies may overrule this parameter. If the optional period parameter is not present, the operator's policy on balance expiration is always in effect.

8.1.3.1 Input message : BalanceUpdateRequest

Part Name	Part Type	Description
EndUserIdentifier	xsd:anyURI	This parameter identifies the end user's account.
EndUserPin	xsd:string	OPTIONAL. Contains the end user's credentials for authorizing access to the account.
ReferenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes
Amount	xsd:decimal	Currency amount that should be added to the end user's account.
Period	xsd:int	OPTIONAL. The balance is requested to expire in the number of days indicated by this parameter. The operator's policies may overrule this parameter. If this optional parameter is not present, the operator's policy on balance expiration is always in effect.

8.1.3.2 Output message : BalanceUpdateResponse

Part Name	Part Type	Description
None.		

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error

- SVC0002: Invalid input value
- SVC0250: End user authentication failed

PolicyException from [6]

- POL0001 – Policy error

8.1.4 Operation : VoucherUpdate

This message results in directly recharging the account indicated by the end user identifier and optional associated end user PIN. The reference code is used to uniquely identify the request; it is the application's responsibility to provide a unique reference code within the scope of the application. A voucher identifier indirectly specifies the charge. The optional voucher PIN code can be used to verify the voucher.

8.1.4.1 Input message : VoucherUpdateRequest

Part Name	Part Type	Description
EndUserIdentifier	xsd:anyURI	This parameter identifies the end user's account.
EndUserPin	xsd:string	OPTIONAL. Contains the end user's credentials for authorizing access to the account.
ReferenceCode	xsd:string	Textual information to uniquely identify the request, e.g. in case of disputes
VoucherIdentifier	xsd:string	This parameter identifies the voucher.
VoucherPin	xsd:string	OPTIONAL. Contains the voucher's credentials for authentication.

8.1.4.2 Output message : VoucherUpdateResponse

Part Name	Part Type	Description
None.		

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0250: End user authentication failed
- SVC0251: Unknown voucher

PolicyException from [6]

- POL0001 – Policy error
- POL0220 – Vouchers not accepted

8.1.5 Operation : GetHistory

This message results in returning the transaction history of the account indicated by the end user identifier and associated optional end user PIN. The maximum number of entries to return and the start date define the range of transactions that are of interest to the requester.

If the total number of entries in the transaction history, starting at the specified date, is larger than the specified maximum number of entries, only the most recent events are returned. Note that the operator might limit the maximum amount of entries to be returned or the period for which the entries are to be returned.

8.1.5.1 Input message : GetHistoryRequest

Part Name	Part Type	Description
EndUserIdentifier	xsd:anyURI	This parameter identifies the end user's account.
EndUserPin	xsd:string	OPTIONAL. Contains the end user's credentials for authorizing access to the account.
Date	xsd:dateTime	OPTIONAL. This parameter indicates the desired starting date for the entries to be returned. If this parameter is not present, it is up to the discretion of the service to decide this date.
MaxEntries	xsd:int	OPTIONAL. This parameter indicates the maximum number of entries that shall be returned. If this parameter is not present, it is up to the discretion of the service to decide how many entries to return.

8.1.5.2 Output message : GetHistoryResponse

Part Name	Part Type	Description
History	DatedTransaction [0 .. unbounded]	It is a DatedTransaction array that consists of types with a date field and a string field: i.e. the date of the occurrence and the transaction details, respectively.

8.1.5.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001 – Policy error

9 Fault Definitions

9.1 Fault : ServiceException

9.1.1 End user authentication failed.

Message Id	<SVC0250>
Text	End user authentication failed.
Variables	None.

9.1.2 Unknown Voucher

Message Id	<SVC0251>
Text	Voucher %1 is not valid.
Variables	%1 Voucher identifier.

9.2 Fault : PolicyException

9.2.1 Vouchers not accepted

Message Id	<POL0220>
Text	Vouchers not accepted.
Variables	None.

10 Service Policies

The following service policies are defined for this service.

Name	Type	Description
VouchersAccepted	xsd:boolean	Are vouchers accepted

Annex A (normative): WSDL for Account Management

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-07-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-8 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 8: Terminal Status
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 7
6 Sequence Diagrams 7
6.1 Terminal Status Query 7
6.2 Terminal Status Group Query 8
6.3 Terminal Status Notification 8
6.4 Terminal Status Notification with Check Immediate 9
7 XML Schema Data Type Definition 11
7.1 Status Enumeration 11
7.2 RetrievalStatus Enumeration 11
7.3 StatusData Structure 11
8 Web Service Interface Definition 12
8.1 Interface : TerminalStatus 12
8.1.1 Operation : GetStatus 12
8.1.1.1 Input message : GetStatusRequest 12
8.1.1.2 Output message : GetStatusResponse 12
8.1.1.3 Referenced Faults 12
8.1.2 Operation : GetStatusForGroup 12
8.1.2.1 Input message : GetStatusForGroupRequest 12
8.1.2.2 Output message : GetStatusForGroupResponse 13
8.1.2.3 Referenced Faults 13
8.2 Interface : TerminalStatusNotificationManager 13
8.2.1 Operation : StartNotification 13
8.2.1.1 Input message : StartNotificationRequest 13
8.2.1.2 Output message : StartNotificationResponse 14
8.2.1.3 Referenced Faults 14
8.2.2 Operation : EndNotification 14
8.2.2.1 Input message : EndNotificationRequest 15
8.2.2.2 Output message : EndNotificationResponse 15
8.2.2.3 Referenced Faults 15
8.3 Interface : TerminalNotification 15
8.3.1 Operation : StatusNotification 15
8.3.1.1 Input message : StatusNotificationRequest 15
8.3.1.2 Output message : StatusNotificationResponse 15
8.3.1.3 Referenced Faults 15
8.3.2 Operation : StatusError 15
8.3.2.1 Input message : StatusErrorRequest 16
8.3.2.2 Output message : StatusErrorResponse 16
8.3.2.3 Referenced Faults 16
8.3.3 Operation : StatusEnd 16
8.3.3.1 Input message : StatusEndRequest 16
8.3.3.2 Output message : StatusEndResponse 16
8.3.3.3 Referenced Faults 16
9 Fault Definitions 16
9.1 Fault : PolicyException 16

10 Service Policies..... 17

Annex A (normative): WSDL for Terminal Status 18

Annex B (informative): Change history..... 19

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 8 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	**Terminal Status**
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 8 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Terminal Status Web Service aspects of the interface. All aspects of the Terminal Status Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

Terminal Status provides access to the status of a terminal through,

- Request for the status of a terminal
- Request for the status of a group of terminals
- Notification of a change in the status of a terminal

The status of a terminal can be expressed as reachable, unreachable or busy – however not all terminals distinguish a busy status, so applications should be able to adapt to what information is available (using the service properties to determine available information).

When a request for a group of terminals is made, the response may contain a full or partial set of results. This allows the service to provide results based on a number of criteria including number of terminals for which the request is made and amount of time required to retrieve the information. This allows the requester to initiate additional requests for those terminals for which information was not provided.

5 Namespaces

The data types are defined in the namespace

www.csapi.org/schema/parlayx/terminal_status/v2_0

The TerminalStatus interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_status/v2_0

The TerminalStatusNotificationManager interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_status/notification_manager/v2_0

The TerminalStatusNotification interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_status/notification/v2_0

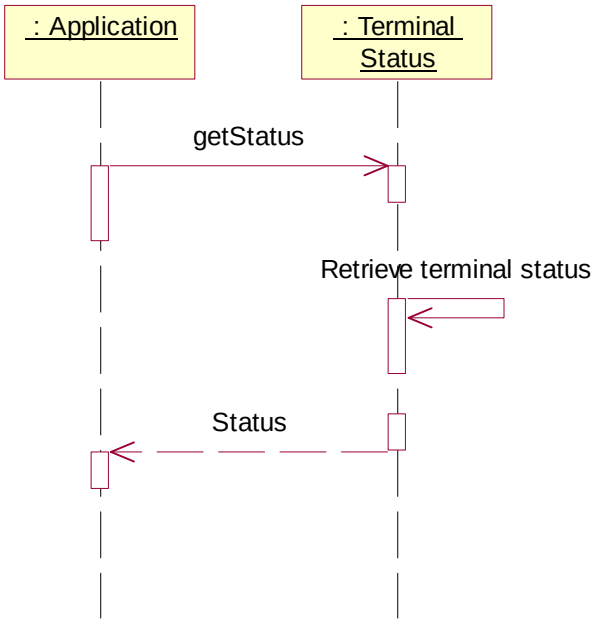
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Terminal Status Query

Pattern: Request / Response

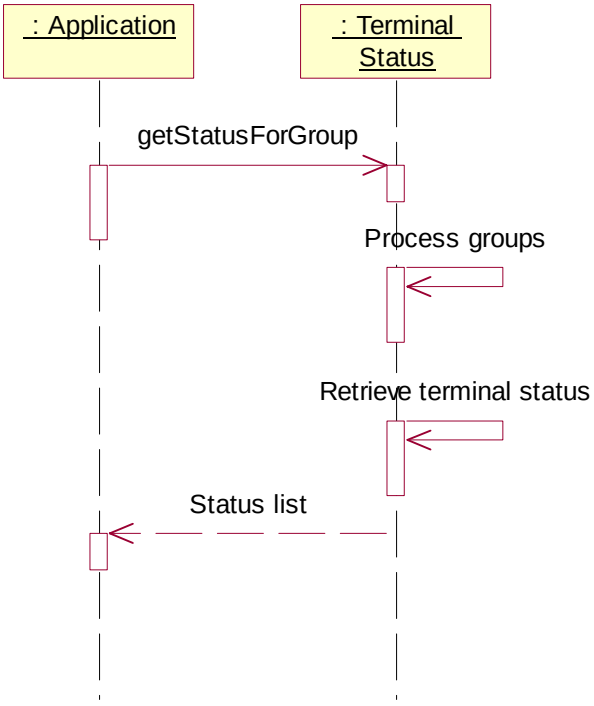
When an application is interested in determining the status of a terminal device, it may provide a terminal device address, and receive the status for the device requested.



6.2 Terminal Status Group Query

Pattern: Request / Response

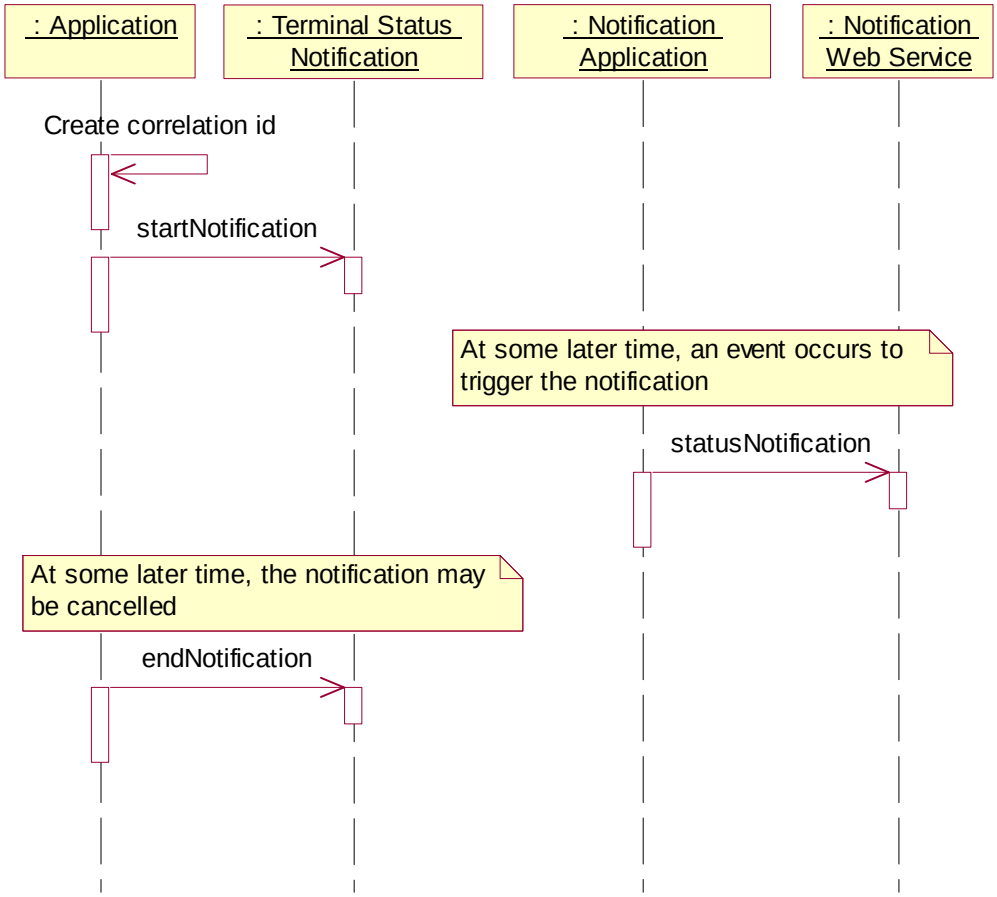
When an application is interested in determining the status of a set of terminal devices, it may provide an array of terminal device addresses, including network managed group addresses, and receive the status for the set of devices requested.



6.3 Terminal Status Notification

Pattern: Application Correlated Multiple Notification

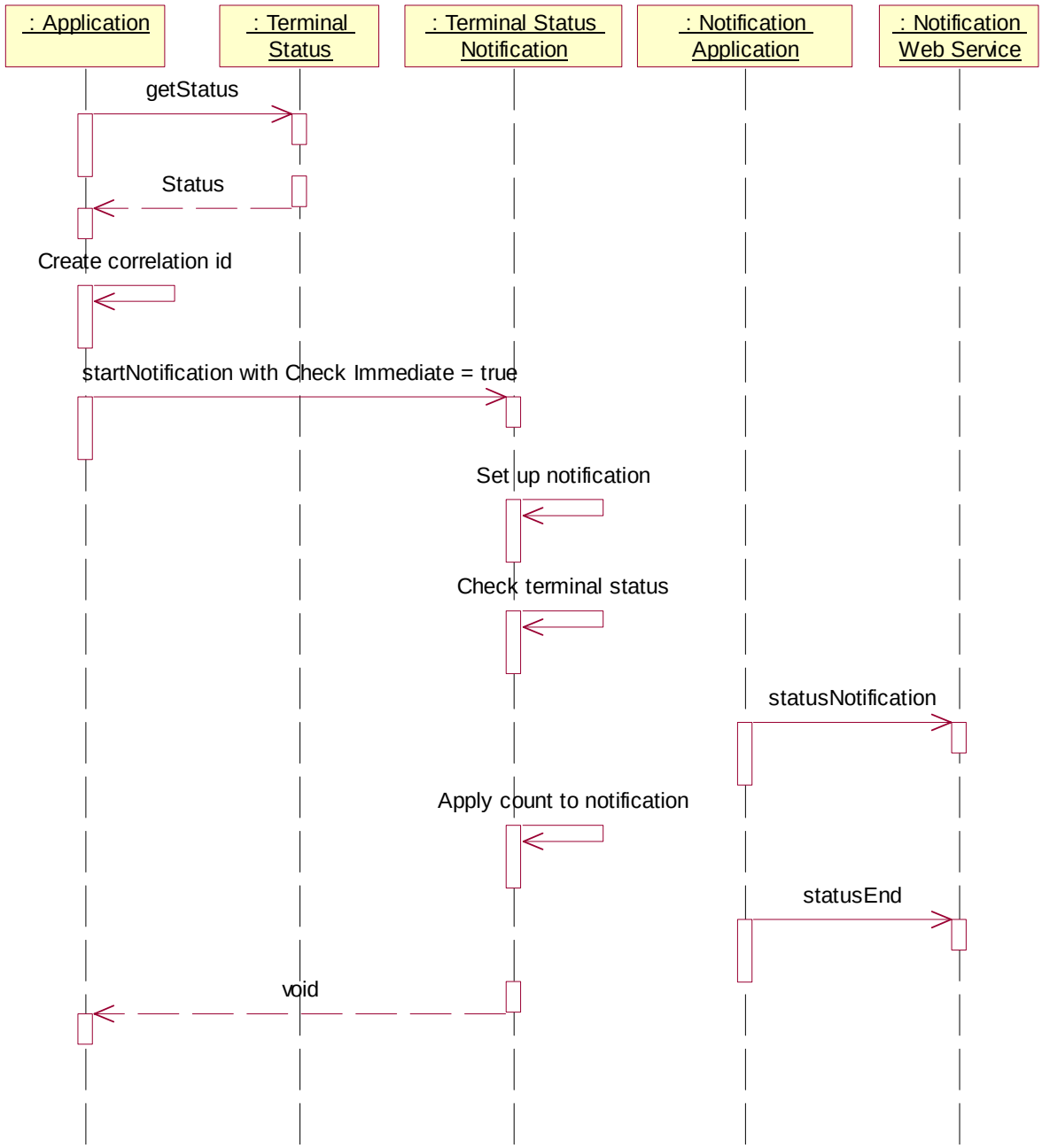
An application can be notified of a change in the status of terminal devices. When the status of a terminal device changes, a notification message will be sent to the application.



6.4 Terminal Status Notification with Check Immediate

In some applications, the terminal status notification will be used to watch for a specific status change. An example is a 'call when available' service, where the terminal status is checked and determined to be not reachable or busy, and a notification is set up to notify the application when the terminal becomes reachable. Between the time that the original status determination and the time the notification is set up, the terminal status could change to reachable – thus the notification on change to reachable would not be sent.

Using the check immediate flag, after the notification is established, the value of the terminal status will be determined, and if the criteria is matched then a notification will be sent immediately. The following sequence diagram shows this scenario.



This sequence shows,

- The Enterprise Application checks the status of a terminal, and receives its status (in this scenario receiving Unreachable or Busy.
- The Enterprise Application generates a correlator, and starts a notification with criteria defined to notify the Enterprise Web Service when the terminal state becomes Reachable and the check immediate flag set to true.
- Sets up the notification to monitor terminal status changes.
- Check the current status of the terminal, and determine if the status matches the criteria.
- In this case, the criteria matches, and a notification is delivered to the Enterprise Web Service
- The count of notifications is incremented and compared to the notification count limit.

- In this case, a single notification was requested, and the end notification message is sent
- The startNotification operation completes

This scenario includes the full set of interactions in one sequence, which also shows that the notifications can be received concurrent with the creation of the notification.

7 XML Schema Data Type Definition

7.1 Status Enumeration

List of possible status values.

Enumeration	Description
Reachable	Terminal is reachable
Unreachable	Terminal is not reachable
Busy	Terminal is busy

7.2 RetrievalStatus Enumeration

Enumeration of the status items that are related to an individual retrieval in a set.

Enumeration	Description
Retrieved	Status retrieved, with result in CurrentStatus element.
NotRetrieved	Status not retrieved, CurrentStatus is not provided (does not indicate an error, no attempt may have been made).
Error	Error retrieving status.

7.3 StatusData Structure

Data structure containing device identifier and its status. As this can be related to a query of a group of terminal devices, the ResultStatus element is used to indicate whether the information for the device was retrieved or not, or if an error occurred.

Name	Type	Description
Address	xsd:anyURI	Address of the Terminal Device to which the status information applies.
ReportStatus	RetrievalStatus	Status of retrieval for this address.
CurrentStatus	Status	Status of terminal.
ErrorInformation	common:ServiceError	If ReportStatus is Error, this is the reason for the error. Error due to privacy verification will be expressed as POL0002 in the ServiceError.

8 Web Service Interface Definition

8.1 Interface : TerminalStatus

Request the status for a terminal or set of terminals.

8.1.1 Operation : GetStatus

This operation is intended to retrieve the status for a single terminal. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.1.1 Input message : GetStatusRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Terminal to request status for

8.1.1.2 Output message : GetStatusResponse

Part Name	Part Type	Description
Result	Status	Status for the terminal for which status was requested

8.1.1.3 Referenced Faults

ServiceException from [6],

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6],

- POL0001: Policy error
- POL0002: Privacy error
- POL0006: Groups not allowed

8.1.2 Operation : GetStatusForGroup

The getStatusForGroup operation initiates a retrieval activity, where one or more terminals, or groups of terminals, may have their status determined.

The Web Service may return a result set that does not include complete information, allowing the Web Service implementation to choose to deliver a partial set of results to accommodate other conditions, such as avoiding timeouts. In this case, the addresses for which no attempt was made to provide data will be marked NotRetrieved in the result for each address this applies to.

8.1.2.1 Input message : GetStatusForGroupRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	List of URIs to get status for, including group URIs

8.1.2.2 Output message : GetStatusForGroupResponse

Part Name	Part Type	Description
Result	StatusData [0..unbounded]	Set of results for the request

8.1.2.3 Referenced Faults

ServiceException from [6],

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses
- SVC0006: Invalid group

PolicyException from [6],

- POL0001: Policy error
- POL0003: Too many addresses
- POL0006: Groups not allowed
- POL0007: Nested groups not allowed

8.2 Interface : TerminalStatusNotificationManager

Set up notifications for terminal status changes.

8.2.1 Operation : StartNotification

Notifications of status changes are made available to applications. The number and duration of notifications may be requested as part of the setup of the notification or may be governed by service policies, or a combination of the two.

If CheckImmediate is set to true, then the notification will be set up, and then the current value of the terminal status will be checked. If the terminal status meets the criteria provided, a notification will be sent to the application. This notification will count against the count requested. This addresses the case where the status of the device changes during the time the notification is being set up, which may be appropriate in some applications.

The correlator provided in the reference must be unique for this Web Service at the time the notification is initiated, otherwise a ServiceException (SVC0005) will be returned to the application.

If the frequency requested is more often than allowed by the service policy, then the value in the service policy will be used. If the duration requested exceeds the time allowed in the service policy, then the value in the service policy will be used. If the notification period (duration) ends before all of the notifications (count) have been delivered, then the notification terminates. In all cases, when the notifications have run their course (by duration or count), an end of notifications message will be provided to the application.

Service policies may govern what count values can be requested, including maximum number of notifications allowed and whether unlimited notifications can be requested (specifying a count of zero). If the count value provided is not in policy, a PolicyException (POL0004 or POL0005 as appropriate) will be returned.

8.2.1.1 Input message : StartNotificationRequest

Part Name	Part Type	Description
Reference	common:SimpleReference	Notification endpoint definition

Addresses	xsd:anyURI [0..unbounded]	Addresses of terminals to monitor
Criteria	Status [0..unbounded]	List of status values to generate notifications for (these apply to all addresses specifed)
CheckImmediate	xsd:boolean	Check status immediately after establishing notification
Frequency	common:TimeMetric	Maximum frequency of notifications (can also be considered minimum time between notifications)
Duration	common:TimeMetric	Length of time notifications occur for, null to use default notification time defined by service policy
Count	xsd:integer	Maximum number of notifications, zero if no maximum

8.2.1.2 Output message : StartNotificationResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

ServiceException from [6],

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses
- SVC0005: Duplicate correlator
- SVC0006: Invalid group

PolicyException from [6],

- POL0001: Policy error
- POL0003: Too many addresses
- POL0004: Unlimited notifications not supported
- POL0005: Too many notifications requested
- POL0006: Groups not allowed
- POL0007: Nested groups not allowed
- POL0009: Invalid frequency requested
- POL0200: Busy criteria not supported

8.2.2 Operation : EndNotification

The application may end a notification using this operation. Until this operation returns, notifications may continue to be received by the application.

An end of notification (statusEnd) message will not be delivered to the application for a notification ended using this operation.

8.2.2.1 Input message : EndNotificationRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator of request to end

8.2.2.2 Output message : EndNotificationResponse

Part Name	Part Type	Description
None		

8.2.2.3 Referenced Faults

ServiceException from [6],

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6],

- POL0001: Policy error

8.3 Interface : TerminalNotification

Notification interface to which notifications are delivered.

8.3.1 Operation : StatusNotification

When the status of a monitored device changes, a notification is delivered to the application with the new status information. If a group identifier was used, the terminal device URI is provided, not the group URI.

8.3.1.1 Input message : StatusNotificationRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification
Address	xsd:anyURI	Address of the terminal the status change applies to
CurrentStatus	Status	New terminal status

8.3.1.2 Output message : StatusNotificationResponse

Part Name	Part Type	Description
None		

8.3.1.3 Referenced Faults

None.

8.3.2 Operation : StatusError

The status changed error message is sent to the application to indicate that the notification is being cancelled by the Web Service.

8.3.2.1 Input message : StatusErrorRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification.
Address	xsd:anyURI	Address of terminal if the error applies to an individual terminal, or nil if applies to whole notification.
Reason	common:ServiceError	Reason notification is being discontinued.

8.3.2.2 Output message : StatusErrorResponse

Part Name	Part Type	Description
None		

8.3.2.3 Referenced Faults

None.

8.3.3 Operation : StatusEnd

The notifications have completed for this correlator. This message will be delivered when the duration or count for notifications have been completed. This message will not be delivered in the case of an error ending the notifications or deliberate ending of the notifications (using endNotification operation).

8.3.3.1 Input message : StatusEndRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification.

8.3.3.2 Output message : StatusEndResponse

Part Name	Part Type	Description
None		

8.3.3.3 Referenced Faults

None.

9 Fault Definitions

New fault definitions for this service.

9.1 Fault : PolicyException

Busy criteria not supported.

Message Id	<POL0200>
Text	Busy criteria is not supported.
Variables	None.

10 Service Policies

Name	Type	Description
BusyAvailable	xsd:boolean	Can busy be returned as a status or be a trigger
MaximumNotificationAddresses	xsd:int	Maximum number of addresses for which a notification can be set up
MaximumNotificationFrequency	common:TimeMetric	Maximum rate of notification delivery (also can be considered minimum time between notifications)
MaximumNotificationDuration	common:TimeMetric	Maximum amount of time a notification may be set up for
MaximumCount	xsd:int	Maximum number of notifications that may be requested
UnlimitedCountAllowed	xsd:boolean	Allowed to specify unlimited notification count (i.e. specify zero in notification count requested)
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions

Annex A (normative): WSDL for Terminal Status

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-08-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-9 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services; Part 9: Terminal Location (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 7
6 Sequence Diagrams 7
6.1 Terminal Location Query 7
6.2 Terminal Location Group Query 8
6.3 Terminal Location Notification 8
6.4 Terminal Location Notification with Check Immediate 9
6.5 Terminal Location Periodic Notification 11
7 XML Schema Data Type Definition 11
7.1 Latitude and Longitude Values 11
7.2 Accuracy Values 12
7.3 EnteringLeavingCriteria Enumeration 12
7.4 LocationInfo Structure 12
7.5 RetrievalStatus Enumeration 12
7.6 LocationData Structure 13
8 Web Service Interface Definition 13
8.1 Interface : TerminalLocation 13
8.1.1 Operation : GetLocation 13
8.1.1.1 Input message : GetLocationRequest 13
8.1.1.2 Output message : GetLocationResponse 13
8.1.1.3 Referenced Faults 13
8.1.2 Operation : GetTerminalDistance 14
8.1.2.1 Input message : GetTerminalDistanceRequest 14
8.1.2.2 Output message : GetTerminalDistanceResponse 14
8.1.2.3 Referenced Faults 14
8.1.3 Operation : GetLocationForGroup 14
8.1.3.1 Input message : GetLocationForGroupRequest 15
8.1.3.2 Output message : GetLocationForGroupResponse 15
8.1.3.3 Referenced Faults 15
8.2 Interface : TerminalLocationNotificationManager 15
8.2.1 Operation : StartGeographicalNotification 15
8.2.1.1 Input message : StartGeographicalNotificationRequest 16
8.2.1.2 Output message : StartGeographicalNotificationResponse 16
8.2.1.3 Referenced Faults 16
8.2.2 Operation : StartPeriodicNotification 17
8.2.2.1 Input message : StartPeriodicNotificationRequest 17
8.2.2.2 Output message : StartPeriodicNotificationResponse 17
8.2.2.3 Referenced Faults 17
8.2.3 Operation : EndNotification 18
8.2.3.1 Input message : EndNotificationRequest 18
8.2.3.2 Output message : EndNotificationResponse 18
8.2.3.3 Referenced Faults 18
8.3 Interface : TerminalLocationNotification 18
8.3.1 Operation : LocationNotification 18
8.3.1.1 Input message : LocationNotificationRequest 18
8.3.1.2 Output message : LocationNotificationResponse 19
8.3.1.3 Referenced Faults 19
8.3.2 Operation : LocationError 19

8.3.2.1 Input message : LocationErrorRequest 19
8.3.2.2 Output message : LocationErrorResponse 19
8.3.2.3 Referenced Faults 19
8.3.3 Operation : LocationEnd 19
8.3.3.1 Input message : LocationEndRequest 19
8.3.3.2 Output message : LocationEndResponse 19
8.3.3.3 Referenced Faults 19
9 Fault Definitions 20
9.1 Fault: ServiceException 20
9.1.1 SVC0200: Accuracy out of limit 20
9.2 Fault : PolicyException 20
9.2.1 POL0230: Requested accuracy not supported 20
9.2.2 POL0231: Geographic notification not available 20
9.2.3 POL0232: Periodic notification not available 20
10 Service Policies 20
Annex A (normative): WSDL for Terminal Location 22
Annex B (informative): Change history 23

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 9 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	**Terminal Location**
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 9 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Terminal Location Web Service aspects of the interface. All aspects of the Terminal Location Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

Terminal Location provides access to the location of a terminal through,

Request for the location of a terminal

Request for the location of a group of terminals

Notification of a change in the location of a terminal

Notification of terminal location on a periodic basis

Location is expressed through a latitude, longitude, altitude and accuracy.

When a request for a group of terminals is made, the response may contain a full or partial set of results. This allows the service to provide results based on a number of criteria including number of terminals for which the request is made and amount of time required to retrieve the information. This allows the requester to initiate additional requests for those terminals for which information was not provided.

5 Namespaces

The Terminal Location interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_location/v2_0

The TerminalLocationNotificationManager interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_location/notification_manager/v2_0

The TerminalLocationNotification interface uses the namespace

www.csapi.org/wsdl/parlayx/terminal_location/notification/v2_0

The data types are defined in the namespace

www.csapi.org/schema/parlayx/terminal_location/v2_0

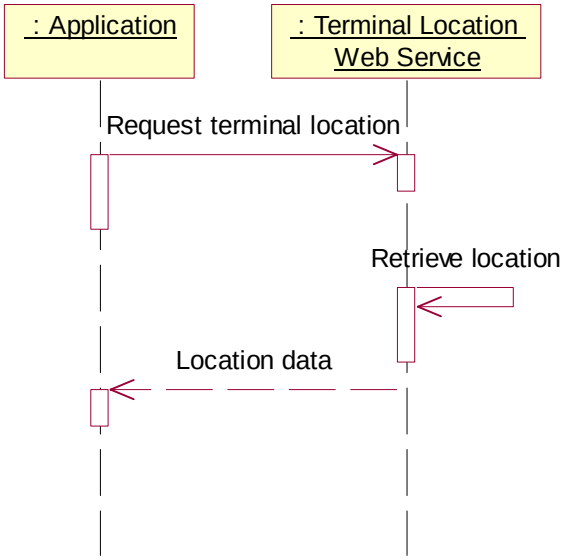
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Terminal Location Query

Pattern: Request / Response

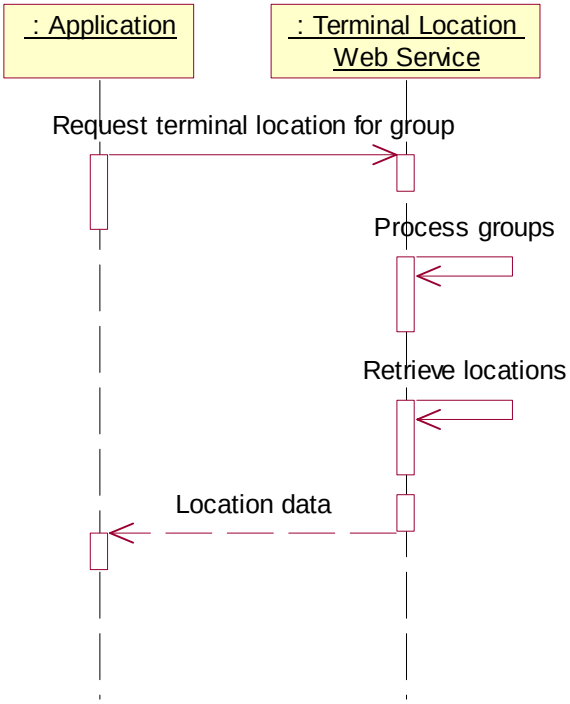
For an application to determine the location of a terminal device, it provides a terminal device address and desired accuracy, and receives the location for the device requested.



6.2 Terminal Location Group Query

Pattern: Request / Response

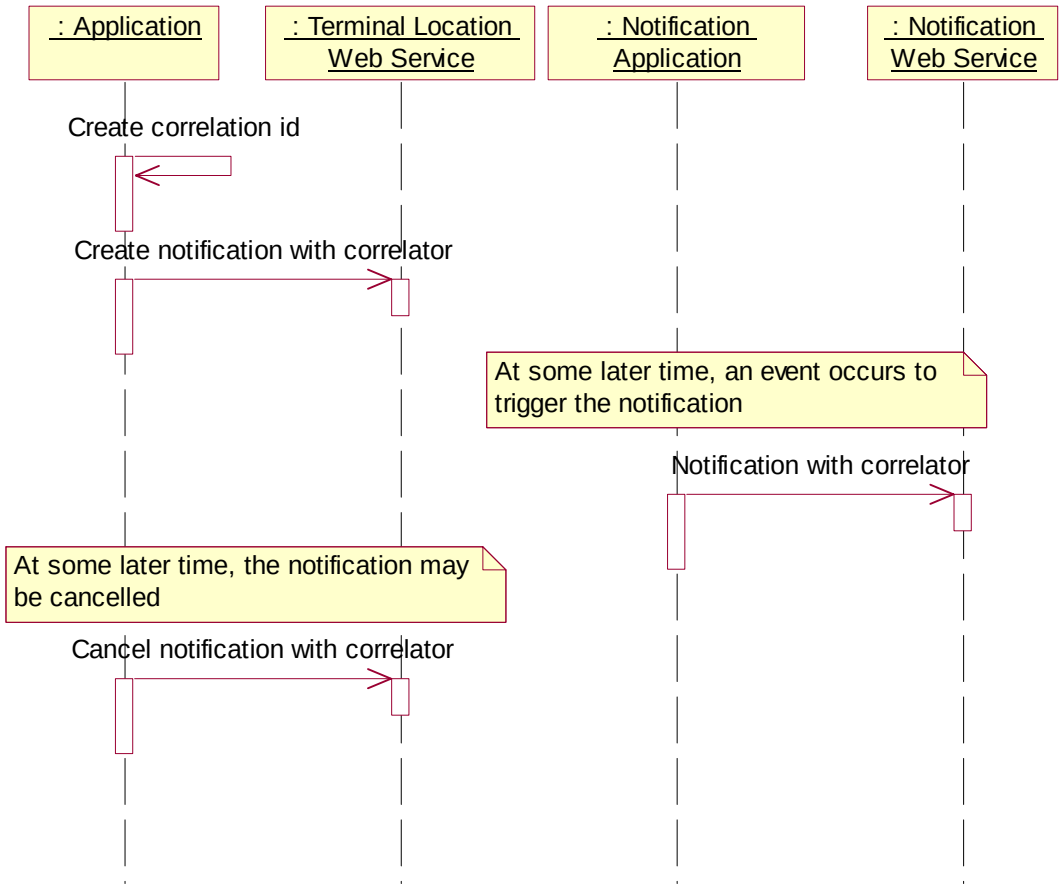
When an application requires the locations of a set of terminal devices, it may provide an array of terminal device addresses, including network managed group addresses, and receive the location data for the set of devices requested.



6.3 Terminal Location Notification

Pattern: Application Correlated Multiple Notification

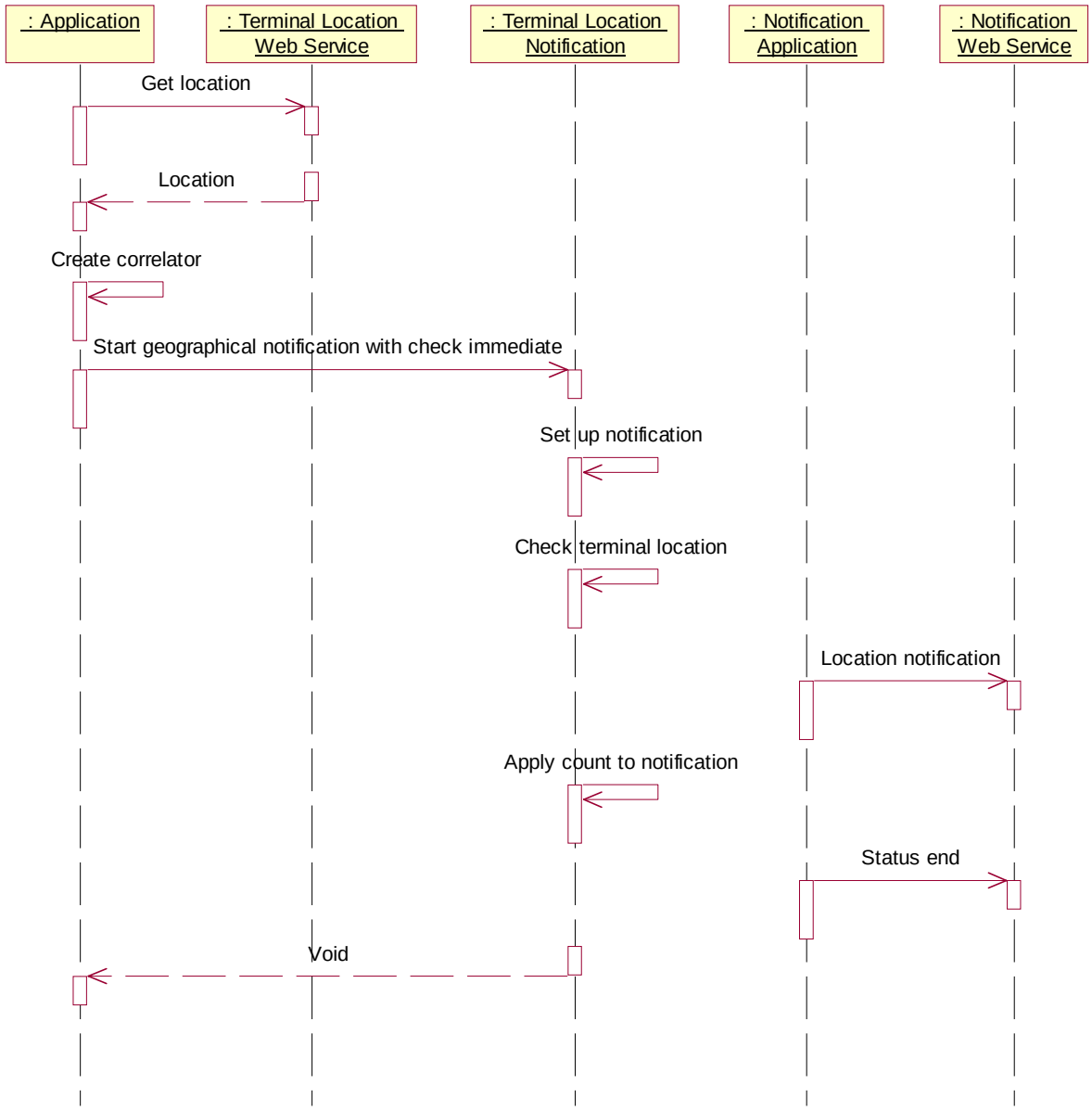
An application can be notified of a terminal device entering or leaving a geographical area. When a matching event occurs; a notification message will be sent to the application.



6.4 Terminal Location Notification with Check Immediate

In some applications, the terminal location notification will be used to watch for a specific location change. An example is a ‘call when present’ service, where the terminal location is checked and determined to be outside the target area, and a notification is set up to notify the application when the terminal enters the target area. Between the time of the original location determination and the time the notification is set up, the terminal could move into the target area – thus the notification on entry into the target area would not be sent.

Using the check immediate flag, after the notification is established, the terminal location will be determined, and if the terminal is in the target area, then a notification will be sent immediately. The following sequence diagram shows this scenario.



This sequence shows,

The Enterprise Application checks the location of a terminal, and receives its location (in this scenario determining that the terminal is outside the target area).

The Enterprise Application generates a correlator, and starts a notification with criteria defined to notify the Enterprise Web Service when the terminal enters the target area and the check immediate flag set to true.

Sets up the notification to monitor terminal location.

Check the current location of the terminal, and determine if the terminal lies inside the target area.

In this case, the terminal is in the target area, and a notification is delivered to the Enterprise Web Service

The count of notifications is incremented and compared to the notification count limit.

In this case, a single notification was requested, and the end notification message is sent

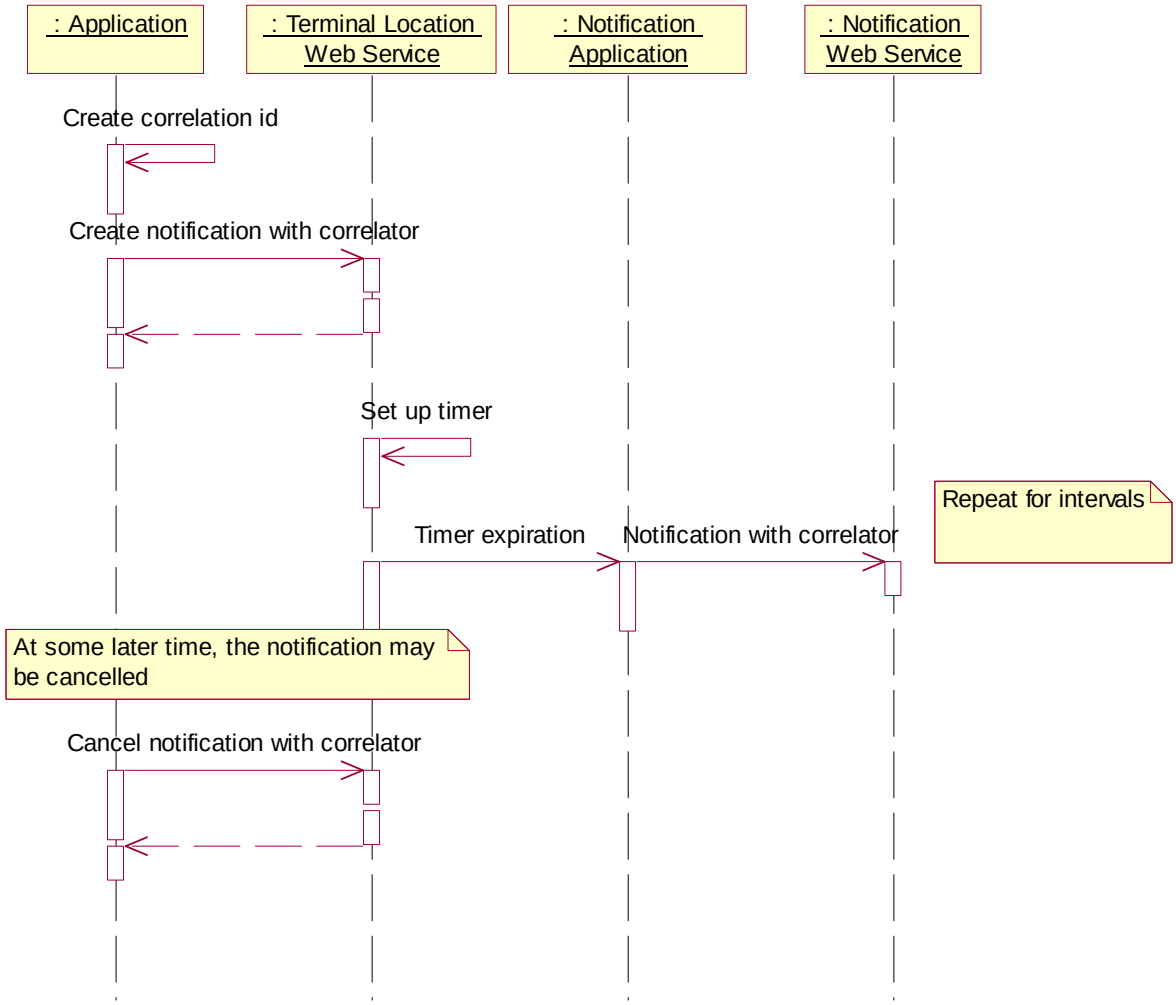
The startGeographicalNotification operation completes

This scenario includes the full set of interactions in one sequence, which also shows that the notifications can be received concurrent with the creation of the notification.

6.5 Terminal Location Periodic Notification

Pattern: Application Correlated Multiple Notification

An application can be notified of a terminal device location on a periodic basis. At each interval, a notification message will be sent to the application.



7 XML Schema Data Type Definition

7.1 Latitude and Longitude Values

Latitude and longitude values used in this specification follow the conventions of the ISO 6709 specification, as it applies to latitudes and longitudes specified using decimal degrees.

Latitude values are expressed as floating point numbers in the range -90.0000 to +90.0000, using decimal degrees (as opposed to minutes and seconds). Positive values indicate locations north of and on the equator. Negative values indicate locations south of the equator.

Longitude values are expressed as floating point numbers in the range -180.0000 to +180.0000, using decimal degrees (as opposed to minutes and seconds). Positive values indicate locations east of and on the prime meridian (Greenwich). Negative values indicate locations west of the prime meridian up to the 180th meridian.

7.2 Accuracy Values

Two accuracy values are used in some of the operations. These values express the desire of the application for the location information to be provided by the Web Service. The choice of values may influence the price that the Service Provider charges.

The 'requested accuracy' expresses the range in which the application wishes to receive location information. This may influence the choice of location technology to use (for instance, cell sector location may be suitable for requests specifying 1000 meters, but GPS technology may be required for requests below 100 meters).

The 'acceptable accuracy' expresses the range that the application considers useful – if the location cannot be determined within this range, then the application would prefer not to receive the information. For instance, a taxi tracking service to determine the closest taxi to a person may not be useful if the accuracy cannot be provided within 1000 meters to provide prompt service. This will also reduce customer satisfaction issues, since results that are not useful can be handled appropriately for billing (e.g. Service Provider may choose not to bill for these).

7.3 EnteringLeavingCriteria Enumeration

Indicator for whether the notification is related to entering an area or leaving an area.

Enumeration	Description
Entering	Terminal is entering an area
Leaving	Terminal is leaving an area

7.4 LocationInfo Structure

Location information represented as a coordinate.

Name	Type	Description
Address	xsd:anyURI	Address of the terminal device to which the location information applies
Latitude	xsd:float	Location latitude
Longitude	xsd:float	Location longitude
Altitude	xsd:float	Location altitude (optional)
Accuracy	xsd:int	Accuracy of location provided in meters
Timestamp	xsd:dateTime	Date and time that location was collected

7.5 RetrievalStatus Enumeration

Enumeration of the location items that are related to an individual retrieval in a set.

Enumeration	Description
Retrieved	Location retrieved, with result in CurrentLocation element.
NotRetrieved	Location not retrieved, CurrentLocation is not provided (does not indicate an error, no attempt may have been made).

Error	Error retrieving location.

7.6 LocationData Structure

Data structure containing device address, retrieval status and location information. As this can be related to a query of a group of terminal devices, the ResultStatus element is used to indicate whether the information for the device was retrieved or not, or if an error occurred.

Name	Type	Description
ReportStatus	RetrievalStatus	Status of retrieval for this terminal device address
CurrentLocation	LocationInfo	Location of terminal
ErrorInformation	common:ServiceError	If report status is error, this is the reason for the error. Error due to privacy verification will be expressed as POL0002 in the ServiceError.

8 Web Service Interface Definition

8.1 Interface : TerminalLocation

Request the location for a terminal.

8.1.1 Operation : GetLocation

This operation is intended to retrieve the location for a single terminal. The accuracy requested is the desired accuracy for the response. The acceptable accuracy is the limit acceptable to the requester. If the accuracy requested cannot be supported, a PolicyException (POL0230) will be returned to the application. If the accuracy of the location is not within the acceptable accuracy limit, then the location will not be returned, instead a ServiceException (SVC0200) will be returned. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.1.1 Input message : GetLocationRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address of the terminal device for which the location information is requested
RequestedAccuracy	xsd:int	Accuracy of location information requested
AcceptableAccuracy	xsd:int	Accuracy that is acceptable for a response

8.1.1.2 Output message : GetLocationResponse

Part Name	Part Type	Description
Result	LocationInfo	Location of the terminal for which location information was requested

8.1.1.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

SVC0200: Accuracy out of limit

PolicyException from [COMMON]

POL0001: Policy error

POL0002: Privacy error

POL0006: Groups not allowed

POL0230: Requested accuracy not supported

8.1.2 Operation : GetTerminalDistance

This operation is intended to determine the distance of a terminal from a location. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.2.1 Input message : GetTerminalDistanceRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address of terminal to check
Latitude	xsd:float	Latitude of the location to measure from
Longitude	xsd:float	Longitude of the location to measure from

8.1.2.2 Output message : GetTerminalDistanceResponse

Part Name	Part Type	Description
Result	xsd:int	Distance from terminal to the location specified in meters

8.1.2.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

PolicyException from [COMMON]

POL0001: Policy error

POL0002: Privacy error

POL0006: Groups not allowed

8.1.3 Operation : GetLocationForGroup

The getLocationForGroup operation initiates a retrieval activity, where one or more terminals, or groups of terminals, may have their locations determined. The accuracy requested is the desired accuracy for the response. If the accuracy requested is not supported, a PolicyException (POL0230) will be returned to the application. If the location retrieved is not within the acceptable accuracy limit, then the location data will contain a ServiceError (SVC0200).

The Web Service may return a result set that does not include complete information, allowing the Web Service implementation to choose to deliver a partial set of results to accommodate other conditions, such as avoiding timeouts. In this case, the addresses for which no attempt was made to provide data will be marked NotRetrieved in the result for each address for which a location retrieved was not attempted.

8.1.3.1 Input message : GetLocationForGroupRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	List of URIs to get location for, including group URIs
RequestedAccuracy	xsd:int	Accuracy of location requested in meters
AcceptableAccuracy	xsd:int	Accuracy that is acceptable for a response in meters

8.1.3.2 Output message : GetLocationForGroupResponse

Part Name	Part Type	Description
Result	LocationData [0..unbounded]	Set of results for the request

8.1.3.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

SVC0004: No valid addresses

SVC0006: Invalid group

PolicyException from [COMMON]

POL0001: Policy error

POL0003: Too many addresses

POL0006: Groups not allowed

POL0007: Nested groups not allowed

POL0230: Requested accuracy not supported

8.2 Interface : TerminalLocationNotificationManager

Set up notifications for terminal location events using geographical based definitions.

8.2.1 Operation : StartGeographicalNotification

Notifications of location changes are made available to applications. The number and duration of notifications may be requested as part of the setup of the notification or may be governed by service policies, or a combination of the two.

If CheckImmediate is set to true, then the notification will be set up, and then the current value of the terminal location will be checked. If the terminal location is within the radius provided and the criteria is Entering or is outside the radius and the criteria is Leaving, a notification will be sent to the application. This notification will count against the count requested. This addresses the case where the location of the device changes during the time the notification is being set up, which may be appropriate in some applications.

The correlator provided in the reference must be unique for this Web Service at the time the notification is initiated, otherwise a ServiceException (SVC0005) will be returned to the application.

If the frequency requested is more often than allowed by the service policy, then the value in the service policy will be used. If the duration requested exceeds the time allowed in the service policy, then the value in the service policy will be used. If the notification period (duration) ends before all of the notifications (count) have been delivered, then the

notification terminates. In all cases, when the notifications have run their course (by duration or count), an end of notifications message will be provided to the application.

Service policies may govern what count values can be requested, including maximum number of notifications allowed and whether unlimited notifications can be requested (specifying a count of zero). If the count value provided is not in policy, a PolicyException (POL0004 or POL0005 as appropriate) will be returned.

8.2.1.1 Input message : StartGeographicalNotificationRequest

Part Name	Part Type	Description
Reference	common:SimpleReference	Notification endpoint definition
Addresses	xsd:anyURI [0..unbounded]	Addresses of terminals to monitor
Latitude	xsd:float	Latitude of center point
Longitude	xsd:float	Longitude of center point
Radius	xsd:float	Radius of circle around center point in meters
Criteria	EnteringLeavingCriteria	Indicates whether the notification should occur when the terminal enters or leaves the target area
CheckImmediate	xsd:boolean	Check location immediately after establishing notification
Frequency	common:TimeMetric	Maximum frequency of notifications (can also be considered minimum time between notifications)
Duration	common:TimeMetric	Length of time notifications occur for, null to use default notification time defined by service policy
Count	xsd:int	Maximum number of notifications, zero if no maximum

8.2.1.2 Output message : StartGeographicalNotificationResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

SVC0004: No valid addresses

SVC0005: Duplicate correlator

SVC0006: Invalid group

PolicyException from [COMMON]

POL0001: Policy error

POL0003: Too many addresses

POL0004: Unlimited notifications not supported

POL0005: Too many notifications requested

POL0006: Groups not allowed

POL0007: Nested groups not allowed

POL0009: Invalid frequency requested

POL0231: Geographic notification not available

8.2.2 Operation : StartPeriodicNotification

Periodic notifications provide location information for a set of terminals at an application defined interval. The accuracy requested is the desired accuracy for the response. If the accuracy requested is not supported, a PolicyException (POL0230) will be returned to the application.

8.2.2.1 Input message : StartPeriodicNotificationRequest

Part Name	Part Type	Description
Reference	common:SimpleReference	Notification endpoint definition
Addresses	xsd:anyURI [0..unbounded]	Addresses of terminals to monitor
RequestedAccuracy	xsd:int	Accuracy of location requested in meters
Frequency	common:TimeMetric	Maximum frequency of notifications (can also be considered minimum time between notifications)
Duration	common:TimeMetric	Length of time notifications occur for, null to use default notification time defined by service policy

8.2.2.2 Output message : StartPeriodicNotificationResponse

Part Name	Part Type	Description
None		

8.2.2.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

SVC0004: No valid addresses

SVC0005: Duplicate correlator

SVC0006: Invalid group

PolicyException from [COMMON]

POL0001: Policy error

POL0003: Too many addresses

POL0006: Groups not allowed

POL0007: Nested groups not allowed

POL0009: Invalid frequency requested

POL0230: Requested accuracy not available

POL0232: Periodic notification not available

8.2.3 Operation : EndNotification

The application may end a notification (either type) using this operation.

Until this operation returns, notifications may continue to be received by the application.

An end of notification (endNotification) message will not be delivered to the application for a notification ended using this operation.

8.2.3.1 Input message : EndNotificationRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator of request to end

8.2.3.2 Output message : EndNotificationResponse

Part Name	Part Type	Description
None		

8.2.3.3 Referenced Faults

ServiceException from [COMMON]

SVC0001: Service error

SVC0002: Invalid input value

PolicyException from [COMMON]

POL0001: Policy error

8.3 Interface : TerminalLocationNotification

Notification interface to which notifications are delivered.

8.3.1 Operation : LocationNotification

When the location of a monitored device changes a notification is delivered to the application with the new location information. If a group identifier was used, the terminal device URI is provided, not the group URI.

8.3.1.1 Input message : LocationNotificationRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification
Data	LocationInfo [0 .. unbounded]	Location information for terminal
Criteria	EnteringLeavingCriteria	Indicates whether the notification was caused by the terminal entering or leaving the target area (provided for geographical notifications, not for periodic notifications)

8.3.1.2 Output message : LocationNotificationResponse

Part Name	Part Type	Description
None		

8.3.1.3 Referenced Faults

None.

8.3.2 Operation : LocationError

The location error message is sent to the application to indicate that the notification for a terminal, or for the whole notification, is being cancelled by the Web Service.

8.3.2.1 Input message : LocationErrorRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification.
Address	xsd:anyURI	Address of terminal if the error applies to an individual terminal, or nil if it applies to the whole notification.
Reason	common:ServiceError	Reason notification is being discontinued.

8.3.2.2 Output message : LocationErrorResponse

Part Name	Part Type	Description
None		

8.3.2.3 Referenced Faults

None.

8.3.3 Operation : LocationEnd

The notifications have completed for this correlator. This message will be delivered when the duration or count for notifications have been completed. This message will not be delivered in the case of an error ending the notifications or deliberate ending of the notifications (using endNotification operation).

8.3.3.1 Input message : LocationEndRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification.

8.3.3.2 Output message : LocationEndResponse

Part Name	Part Type	Description
None		

8.3.3.3 Referenced Faults

None.

9 Fault Definitions

New fault definitions for this service.

9.1 Fault: ServiceException

9.1.1 SVC0200: Accuracy out of limit.

Message Id	SVC0200
Text	Accuracy of location is not within acceptable limit.
Variables	

9.2 Fault : PolicyException

9.2.1 POL0230: Requested accuracy not supported.

Message Id	POL0230
Text	Requested accuracy is not supported.
Variables	None

9.2.2 POL0231: Geographic notification not available

Message Id	POL0231
Text	Geographic notification is not available
Variables	None

9.2.3 POL0232: Periodic notification not available

Message Id	POL0232
Text	Periodic notification is not available
Variables	None

10 Service Policies

Name	Type	Description
MinimumAccuracy	xsd:int	Minimum value for requested accuracy
MinimumAcceptableAccuracy	xsd:int	Minimum value for acceptable accuracy
GeographicalNotificationAvailable	xsd:boolean	Can notifications be set on a geography
PeriodicNotificationAvailable	xsd:boolean	Can a periodic notification be set up
AltitudeAlwaysAvailable	xsd:boolean	Is altitude available for all location responses

AltitudeSometimesAvailable	xsd:boolean	Is altitude available for some or all location responses (if AltitudeAlwaysAvailable is true, this is also true)
MaximumNotificationAddresses	xsd:int	Maximum number of addresses for which a notification can be set up
MaximumNotificationFrequency	common:TimeMetric	Maximum rate of notification delivery (also can be considered minimum time between notifications)
MaximumNotificationDuration	common:TimeMetric	Maximum amount of time a notification may be set up for
MaximumCount	xsd:int	Maximum number of notifications that may be requested
UnlimitedCountAllowed	xsd:boolean	Allowed to specify unlimited notification count (i.e. specify zero in notification count requested)
GroupSupport	xsd:boolean	Groups URIs may be used
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions

Annex A (normative): WSDL for Terminal Location

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-09-200-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Dec 2003	CN_21	NP-030552	--	--	Submitted to CN#22 for Information	1.0.0	
Jan 2004	--	--	--	--	Added The W3C WSDL representation of the APIs specified in the present document is contained in a set of files which accompany the present document: px0326rpcenc.zip px0326rpclit.zip	1.0.1	
Jun 2004	CN_24	NP-040274	--	--	Split into multi-part specification. 29.199-0n, for n=1,2…9. Submitted to CN#24 for Information	1.0.3	
Sep 2004	CN_25	NP-040360	--	--	Draft v200 submitted to TSG CN#25 for Approval.	2.0.0	

3GPP TS 29.199-10 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 10: Call Handling
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 5
2 References 5
3 Definitions and Abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 6
4 Detailed Service Description 6
5 Namespaces 6
6 Sequence Diagrams 7
6.1 Setup Call Handling, Query and Clear Rules 7
7 XML Schema Data Type Definition 9
7.1 ConditionalForward structure 9
7.2 UnconditionalForward structure 9
7.3 InteractionContent enumeration 9
7.4 TextInteraction structure 9
7.5 VoiceInteraction union 9
7.6 CallHandlingRules structure 10
7.7 SetRulesResult structure 10
8 Web Service Interface Definition 10
8.1 Interface : CallHandling 10
8.1.1 Operation : SetRules 10
8.1.1.1 Input message : SetRulesRequest 11
8.1.1.2 Output message : SetRulesResponse 11
8.1.1.3 Referenced Faults 11
8.1.2 Operation : SetRulesForGroup 11
8.1.2.1 Input message : SetRulesForGroupRequest 11
8.1.2.2 Output message : SetRulesForGroupResponse 11
8.1.2.3 Referenced Faults 11
8.1.3 Operation : GetRules 12
8.1.3.1 Input message : GetRulesRequest 12
8.1.3.2 Output message : GetRulesResponse 12
8.1.3.3 Referenced Faults 12
8.1.4 Operation : ClearRules 12
8.1.4.1 Input message : ClearRulesRequest 12
8.1.4.2 Output message : ClearRulesResponse 12
8.1.4.3 Referenced Faults 13
9 Fault Definitions 13
10 Service Policies 13
Annex A (normative): WSDL for Call Handling 14
Annex B (informative): Change history 15

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 10 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	**Call Handling**
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 10 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Call Handling Web Service aspects of the interface. All aspects of the Call Handling Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

The Call Handling Web Service provides a mechanism for an application to specify how calls are to be handled for a specific number. Call handling includes commonly utilized actions,

- Call accepting – only accepting calls from a list of numbers
- Call blocking – blocking calls if they are on a blocking list
- Conditional call forwarding – changing the destination of a call to another number for a specific calling number.
- Unconditional call forwarding – changing the destination of a call to another number
- Play audio – initiate audio with the caller (e.g. an announcement or menu).

The set of rules are provided to the Web Service which is responsible for establishing the call handling function. Only one action is taken for a call, and once this action is started the rules will stop being processed.

There is a specific order in which these rules are processed, providing a predictable call handling expectation for rules provided. The processing is done as follows.

1. Call accepting determines if the call is accepted or rejected. If the caller is not on the accept list, the call is rejected and rule processing ends.
2. Call blocking determines if the call is rejected. If the caller is on the block list, the call is rejected and rule processing ends.
3. Conditional call forwarding – each calling number that has a specific forwarding instruction is checked, and the call is forwarded on a match, and rule processing ends.
4. Unconditional call forwarding – the called number is changed to the call forwarding number and rule processing ends.
5. Play audio – the call is handled by a voice system, which handles all further processing of the call. Rule processing ends when the call is handed off.
6. Continue processing call, to complete call to the original called number.

If no rules are specified in a particular area, then that step is skipped. If the rule processing ends without any action being indicated, then the call will continue to the called number.

Call Handling provides its function without further interaction with the Application. This is in contrast to the Call Notification interfaces which provide notifications to the Application for processing.

5 Namespaces

The Call Handling interface uses the namespace

www.csapi.org/wsdl/parlayx/call handling/v2_0

The data types are defined in the namespace

www.csapi.org/schema/parlayx/call_handling/v2_0

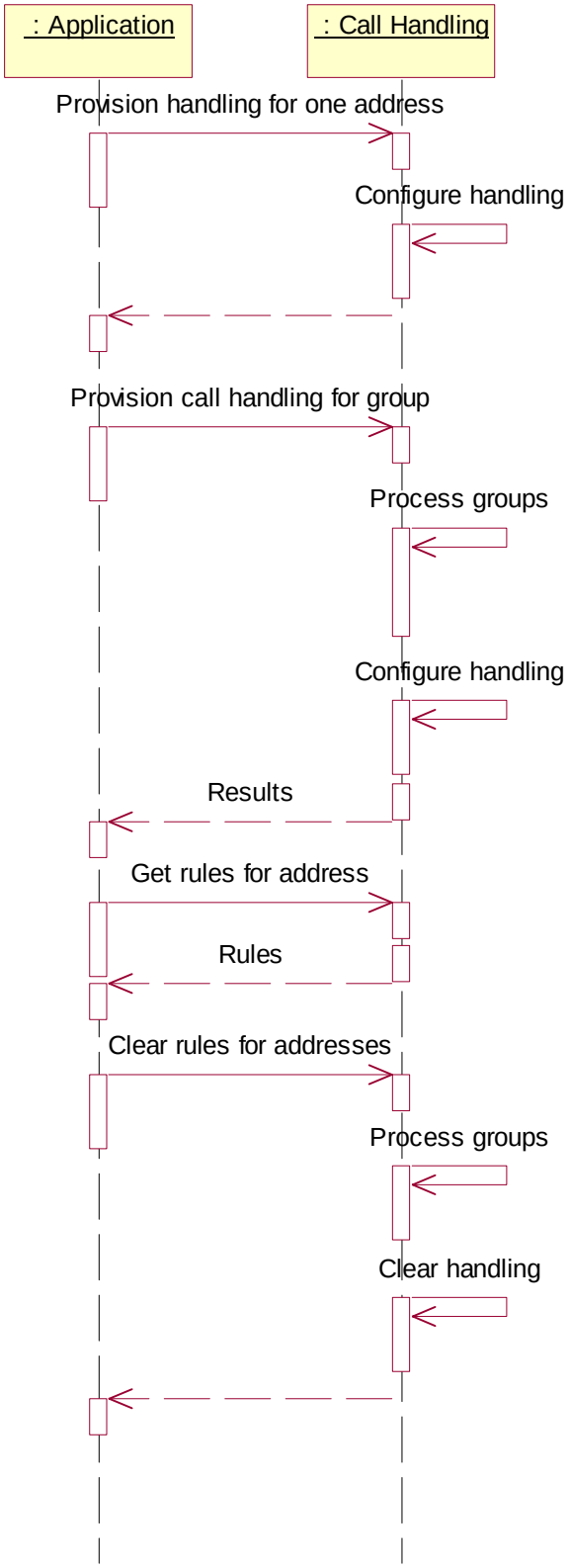
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Setup Call Handling, Query and Clear Rules

Pattern: Request / Response

This sequence shows the application setting up Call Handling with rules to be processed., querying those rules and clearing them.



7 XML Schema Data Type Definition

7.1 ConditionalForward structure

Information on handling of forwarding for specific calling numbers.

Element Name	Element Type	Description
CallingAddress	xsd:anyURI	Address that call is placed from
ForwardingAddress	xsd:anyURI	Address to forward call to
OnBusyAddress	xsd:anyURI	If line is busy at forwarding address, forward to this address
OnNoAnswerAddress	xsd:anyURI	If no answer at forwarding address, forward to this address

7.2 UnconditionalForward structure

Information for handling of forwarding unconditionally.

Element Name	Element Type	Description
ForwardingAddress	xsd:anyURI	Address to forward call to
OnBusyAddress	xsd:anyURI	If line is busy at forwarding address, forward to this address
OnNoAnswerAddress	xsd:anyURI	If no answer at forwarding address, forward to this address

7.3 InteractionContent enumeration

The following are the types of content that may be used for user interaction.

Enumeration	Description
Text	Text to be processed by a Text-To-Speech engine
VoiceXml	VoiceXML to be processed by a VoiceXML browser
Audio	Audio file to be played by an audio processor

7.4 TextInteraction structure

Information for processing by a text to speech engine.

Element Name	Element Type	Description
Text	xsd:string	Text to play through a Text-To-Speech engine
Language	xsd:string	Language of text

7.5 VoiceInteraction union

For a call that is to be handled by an interactive voice system, the information to provide to that system.

Element Name	Element Type	Description

UnionElement	InteractionContent	Type of content provided (one of the following)
TextInfo	TextInteraction	Announcement to play through a Text-To-Speech engine
VoiceXml	xsd:anyURI	Location of VoiceXML to use in a VoiceXML browser
Audio	xsd:anyURI	Location of audio content (WAV or MP3 file)

7.6 CallHandlingRules structure

Structure containing set of rules that are applied when the call is handled

Element Name	Element Type	Description
AcceptList	xsd:anyURI [0..unbounded]	List of addresses to accept calls from
BlockList	xsd:anyURI [0..unbounded]	List of addresses to block calls from
ForwardList	ConditionalForward [0..unbounded]	List of conditional forwarding addresses and destinations
Forward	UnconditionalForward	Unconditional call forwarding address
VoiceInteractionContent	VoiceInteraction	Forward call to a user interaction system with information on content

7.7 SetRulesResult structure

Result of SetRulesRequest for each address.

Element Name	Element Type	Description
Address	xsd:anyURI	Address to be set
Successful	xsd:boolean	Successfully set rules or not
Error	common:ServiceError	Error message if unsuccessful

8 Web Service Interface Definition

8.1 Interface : CallHandling

CallHandling provides a rule based processing capability that is accessible to Applications through a set of operations that allow definition of discrete rules.

8.1.1 Operation : SetRules

Set the call handling rules for an address (the destination for the call). If a set of rules is already in place for any of the **Address**, then this operation will replace the old rules with the set provided in this operation.

The **Address** may not specify a group. If a group is specified, a **PolicyException** will be returned.

8.1.1.1 Input message : SetRulesRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address to handle calls for
Rules	CallHandlingRules	Rules to apply for this address

8.1.1.2 Output message : SetRulesResponse

Part Name	Part Type	Description
None		

8.1.1.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001 : Service error
- SVC0002 : Invalid input value

PolicyException from [COMMON]

- POL0001 : Policy error
- POL0006 : Groups not allowed

8.1.2 Operation : SetRulesForGroup

Set the call handling rules for multiple addresses (the destination for calls). If a set of rules is already in place for any of the **Addresses**, then this operation will replace the old rules with the set provided in this operation.

The **Addresses** may include groups, with members using the 'tel:' and 'sip:' URIs in the manner defined in [6]. Wildcards may not be used to specify addresses.

8.1.2.1 Input message : SetRulesForGroupRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	Addresses to handle calls for
Rules	CallHandlingRules	Rules to apply for these addresses

8.1.2.2 Output message : SetRulesForGroupResponse

Part Name	Part Type	Description
Result	SetRulesResult [0..unbounded]	Result of setup for each of addresses provided

8.1.2.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001 : Service error
- SVC0002 : Invalid input value
- SVC0004 : No valid addresses

- SVC0006 : Invalid group

PolicyException from [COMMON]

- POL0001 : Policy error
- POL0006 : Groups not allowed
- POL0007 : Nested groups not allowed

8.1.3 Operation : GetRules

Get the call handling rules for an address (the destination for the call).

The **Address** may not specify a group. If a group is specified, a **PolicyException** will be returned.

8.1.3.1 Input message : GetRulesRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address to handle calls for

8.1.3.2 Output message : GetRulesResponse

Part Name	Part Type	Description
Rules	CallHandlingRules	Rules being applied for this address

8.1.3.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001 : Service error
- SVC0002 : Invalid input value

PolicyException from [COMMON]

- POL0001 : Policy error
- POL0006 : Groups not allowed

8.1.4 Operation : ClearRules

Clear the call handling rules associated with the addresses specified. If no rules have been set for an address, this operation silently ignores the request, and does not return an error or fault message.

The **Addresses** may include groups, with members using the ‘tel:’ and ‘sip:’ URIs in the manner defined in [6]. Wildcards may not be used to specify addresses.

8.1.4.1 Input message : ClearRulesRequest

Part Name	Part Type	Description
Addresses	xsd:anyURI [0..unbounded]	Addresses to clear call handling for

8.1.4.2 Output message : ClearRulesResponse

Part Name	Part Type	Description

None		

8.1.4.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001 : Service error
- SVC0002 : Invalid input value
- SVC0006 : Invalid group

PolicyException from [COMMON]

- POL0001 : Policy error
- POL0006 : Groups not allowed
- POL0007 : Nested groups not allowed

9 Fault Definitions

No new faults defined for this service.

10 Service Policies

Name	Type	Description
VoiceMailAvailable	xsd:boolean	Voice mail available or not
TextToSpeechAvailable	xsd:boolean	Service accepts text as an input for processing with a Text-To-Speech engine
AudioContentAvailable	xsd:boolean	Service accepts audio content for playing with an audio player
VoiceXmlAvailable	xsd:boolean	Service accepts VoiceXML for processing with a VoiceXML browser
AudioFormatsSupported	xsd:string	Comma separated string of audio formats supported (e.g. WAV,MP3,AU)
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions

Annex A (normative): WSDL for Call Handling

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-10-100-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040360	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	

3GPP TS 29.199-11 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 11: Audio Call
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 5
2 References 5
3 Definitions and Abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 6
4 Detailed Service Description 6
5 Namespaces 6
6 Sequence Diagrams 6
6.1 Play Audio and Check Status 6
6.2 Play Audio and Cancel 7
7 XML Schema Data Type Definition 8
7.1 MessageStatus Enumeration 8
8 Web Service Interface Definition 9
8.1 Interface : PlayAudio 9
8.1.1 Operation : PlayTextMessage 9
8.1.1.1 Input message : PlayTextMessageRequest 9
8.1.1.2 Output message : PlayTextMessageResponse 9
8.1.1.3 Referenced Faults 9
8.1.2 Operation : PlayAudioMessage 10
8.1.2.1 Input message : PlayAudioMessageRequest 10
8.1.2.2 Output message : PlayAudioMessageResponse 10
8.1.2.3 Referenced Faults 10
8.1.3 Operation : PlayVoiceXmlMessage 10
8.1.3.1 Input message : PlayVoiceXmlMessageRequest 10
8.1.3.2 Output message : PlayVoiceXMLMessageResponse 11
8.1.3.3 Referenced Faults 11
8.1.4 Operation : GetMessageStatus 11
8.1.4.1 Input message : GetMessageStatusRequest 11
8.1.4.2 Output message : GetMessageStatusResponse 11
8.1.4.3 Referenced Faults 11
8.1.5 Operation : EndMessage 12
8.1.5.1 Input message : EndMessageRequest 12
8.1.5.2 Output message : EndMessageResponse 12
8.1.5.3 Referenced Faults 12
9 Fault Definitions 12
10 Service Policies 12
Annex A (normative): WSDL for Audio Call 13
Annex B (informative): Change history 14

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 11 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	**Audio Call**
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 11 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Audio Call Web Service aspects of the interface. All aspects of the Audio Call Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

The Audio Call service provides a flexible way to provide vocal message delivery. The interface is very simple, not requiring the developer to manage the creation of the call nor the interactions with the call to deliver the voice message.

There are three mechanisms which may be utilized for the vocal message content,

- Text, to be rendered using a Text-To-Speech (TTS) engine.
- Audio content (such as .WAV content), to be rendered by an audio player
- VoiceXML, to be rendered using a VoiceXML browser

The service may provide one, two or all three mechanisms, with the service policies providing the mechanism for determining which are available.

5 Namespaces

The data types are defined in the namespace

www.csapi.org/schema/parlayx/audio_call/v2_0

The AudioCall interface uses the namespace

www.csapi.org/wsdl/parlayx/audio_call/v2_0

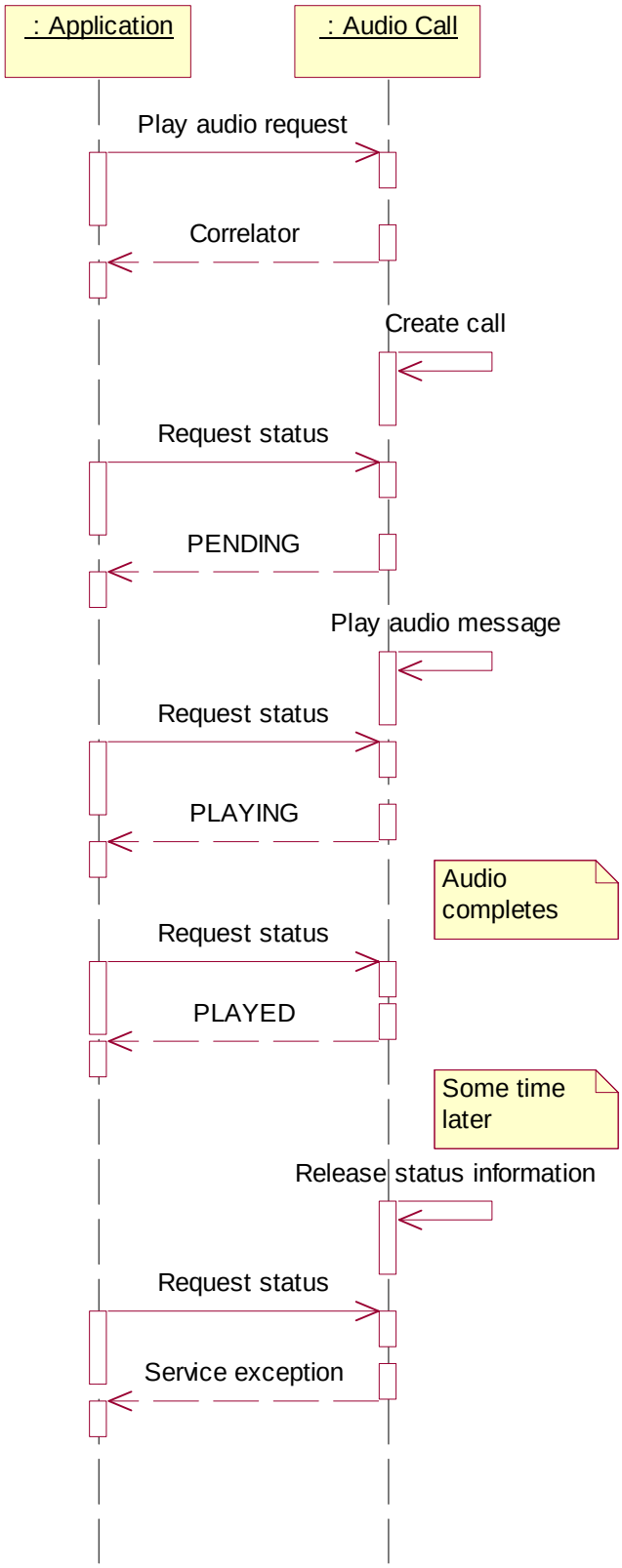
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Play Audio and Check Status

Pattern: Request / response

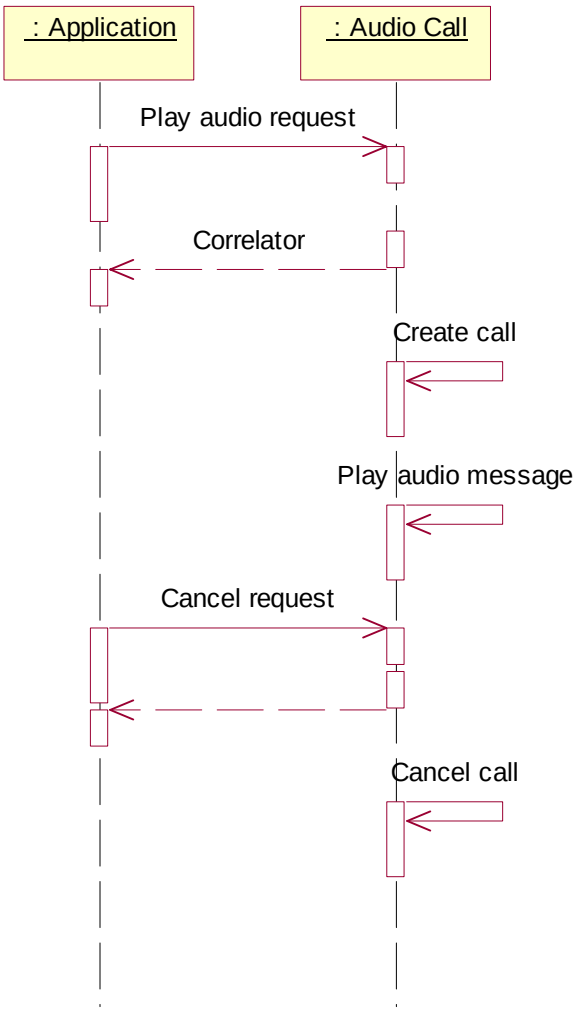
This example shows an audio message being played, and the different responses to status requests that occur at different phases. Note that the last response, a service exception, reflects the transient nature of results, and that these results will expire.



6.2 Play Audio and Cancel

Pattern: Request / response

The playing of a message may be ended by the requester, as shown.



7 XML Schema Data Type Definition

7.1 MessageStatus Enumeration

Status of the message after play message operation has been invoked.

Element Name	Description
Played	Message has been played
Playing	Message is currently playing
Pending	Message has not yet started playing
Error	An error has occurred, message will not be played

8 Web Service Interface Definition

8.1 Interface : PlayAudio

The PlayAudio interface allows the playing of audio messages using different forms of audio content, and operations to monitor or cancel requests.

In all operations, the **Address** is restricted to the use of 'tel:' and 'sip:' URIs as specified in [6], and wildcards are not permitted in these URIs.

8.1.1 Operation : PlayTextMessage

The invocation of **PlayTextMessage** requests to set up a call to the user identified by **Address** and play a text identified by **Text**. The text will be read through a Text-to-Speech engine, according to the specified **Language**. The invocation returns as soon as the request is received by the system, i.e. the actual call is performed asynchronously. The **Correlator**, returned by the invocation, can be used to identify the request, e.g., to get information on the request status.

This operation is intended to play a message to a single terminal. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.1.1 Input message : PlayTextMessageRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address to which message is to be played
Text	xsd:string	Text to process with a Text-To-Speech engine
Language	xsd:string	Language of text (ISO string)
Charging	common:ChargingInformation	Charge to apply for the playing of this message. If charging is not supported then a PolicyException (POL0008) will be returned.

8.1.1.2 Output message : PlayTextMessageResponse

Part Name	Part Type	Description
Correlator	xsd:string	Correlator for this message for subsequent interactions

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0002: Privacy error
- POL0006: Groups not allowed
- POL0008: Charging not supported

8.1.2 Operation : PlayAudioMessage

The invocation of **playAudioMessage** requests to set up a call to the user identified by **Address** and play an audio file located at **AudioUrl**. The invocation returns as soon as the request is received by the system, i.e. the actual call is performed asynchronously. The **Correlator**, returned by the invocation, can be used to identify the request, e.g., to get information on the request status.

This operation is intended to play a message to a single terminal. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.2.1 Input message : PlayAudioMessageRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address to which message is to be played
AudioUrl	xsd:anyURI	Location of audio content to play
Charging	common:ChargingInformation	Charge to apply for the playing of this message. If charging is not supported then a PolicyException (POL0008) will be returned.

8.1.2.2 Output message : PlayAudioMessageResponse

Part Name	Part Type	Description
Correlator	xsd:string	Correlator for this message for subsequent interactions

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0002: Privacy error
- POL0006: Groups not allowed
- POL0008: Charging not supported

8.1.3 Operation : PlayVoiceXmlMessage

The invocation of **PlayVoiceXmlMessage** requests to set up a call to the user identified by **Address** and process VoiceXML content located at **VoiceXmlUrl**. The invocation returns as soon as the request is received by the system, i.e. the actual call is performed asynchronously. The **Correlator**, returned by the invocation, can be used to identify the request, e.g., to get information on the request status.

This operation is intended to play a message to a single terminal. The URI provided is for a single terminal, not a group URI. If a group URI is provided, a PolicyException will be returned to the application.

8.1.3.1 Input message : PlayVoiceXmlMessageRequest

Part Name	Part Type	Description
Address	xsd:anyURI	Address to which message is to be played

VoiceXmlUrl	xsd:anyURI	Location of VoiceXML content to process
Charging	common:ChargingInformation	Charge to apply for the playing of this message. If charging is not supported then a PolicyException (POL0008) will be returned.

8.1.3.2 Output message : PlayVoiceXMLMessageResponse

Part Name	Part Type	Description
Correlator	xsd:string	Correlator for this message for subsequent interactions

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0002: Privacy error
- POL0006: Groups not allowed
- POL0008: Charging not supported

8.1.4 Operation : GetMessageStatus

The invocation of **GetMessageStatus** retrieves the current status, **Result**, of a previous request identified by **Correlator**.

8.1.4.1 Input message : GetMessageStatusRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator returned from play operation to check

8.1.4.2 Output message : GetMessageStatusResponse

Part Name	Part Type	Description
Result	MessageStatus	Current playing status

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.5 Operation : EndMessage

The invocation of **EndMessage** cancels/stops a previous request identified by **Correlator**. It returns a **Result**, with the status of the request at the moment of abort.

8.1.5.1 Input message : EndMessageRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator returned from play operation to cancel

8.1.5.2 Output message : EndMessageResponse

Part Name	Part Type	Description
Result	MessageStatus	Status at the time the endMessage was acted on

8.1.5.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

9 Fault Definitions

No new fault definitions for this service.

10 Service Policies

Name	Type	Description
TextToSpeechAvailable	xsd:boolean	Service accepts text as an input for processing with a Text-To-Speech engine.
AudioContentAvailable	xsd:boolean	Services accepts audio content for playing with an audio player.
VoiceXMLAvailable	xsd:boolean	Service accepts VoiceXML as an input for processing with a VoiceXML browser.
StatusRetainTime	xsd:int	Number of seconds status is retained for after a message is played or an error occurs.
AudioFormatsSupported	xsd:string	Comma separated list of audio formats supported (e.g. WAV, MP3, AU)
ChargingSupported	xsd:boolean	Is charging supported for the play operations

Annex A (normative): WSDL for Audio Call

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-11-100-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040360	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	

3GPP TS 29.199-12 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services; Part 12: Multimedia Conference (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
5 Namespaces 8
6 Sequence Diagrams 8
6.1 Setting Up a Conference 8
6.2 Adding and Removing Media 9
6.3 Conference Owner Disconnects 10
6.4 All Participants Disconnect 11
6.5 Conference Ended by Application 12
7 XML Schema Data Type Definition 13
7.1 ConferenceStatus Enumeration 13
7.2 ConferenceInfo Structure 14
7.3 ParticipantInfo Structure 14
7.4 ParticipantStatus Enumeration 14
7.5 Media Enumeration 14
7.6 MediaDirection Enumeration 15
8 Web Service Interface Definition 15
8.1 Interface : MultimediaConference 15
8.1.1 Operation : createConference 15
8.1.1.1 Input message : createConferenceRequest 15
8.1.1.2 Output message : createConferenceResponse 16
8.1.1.3 Referenced Faults 16
8.1.2 Operation : getConferenceInfo 16
8.1.2.1 Input message : getConferenceInfoRequest 16
8.1.2.2 Output message : getConferenceInfoResponse 16
8.1.2.3 Referenced Faults 17
8.1.3 Operation : endConference 17
8.1.3.1 Input message : endConferenceRequest 17
8.1.3.2 Output message : endConferenceResponse 17
8.1.3.3 Referenced Faults 17
8.1.4 Operation : inviteParticipant 17
8.1.4.1 Input message : inviteParticipantRequest 17
8.1.4.2 Output message : inviteParticipantResponse 17
8.1.4.3 Referenced Faults 18
8.1.5 Operation : disconnectParticipant 18
8.1.5.1 Input message : disconnectParticipantRequest 18
8.1.5.2 Output message : disconnectParticipantResponse 18
8.1.5.3 Referenced Faults 18
8.1.6 Operation : getParticipantInfo 18
8.1.6.1 Input message : getParticipantInfoRequest 18
8.1.6.2 Output message : getParticipantInfoResponse 19
8.1.6.3 Referenced Faults 19
8.1.7 Operation : getParticipants 19
8.1.7.1 Input message : getParticipantsRequest 19
8.1.7.2 Output message : getParticipantsResponse 19
8.1.7.3 Referenced Faults 19
8.1.8 Operation : addMediaForParticipant 19
8.1.8.1 Input message : addMediaForParticipantRequest 19
8.1.8.2 Output message : addMediaForParticipantResponse 20

8.1.8.3 Referenced Faults 20
8.1.9 Operation : deleteMediaForParticipant 20
8.1.9.1 Input message : deleteMediaForParticipantRequest 20
8.1.9.2 Output message : deleteMediaForParticipantResponse 20
8.1.9.3 Referenced Faults 20

9 Fault Definitions 21
9.1 PolicyException 21
9.1.1 POL0240: Too many participants 21
9.1.2 POL0241: Unavailable media 21
9.1.3 POL0242: Maximum duration exceeded 21

10 Service Policies 21

Annex A (normative): WSDL for Multimedia Conference 22

Annex B (informative): Change history 23

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 12 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	**Multimedia Conference**
Part 13:	Address List Management
Part 14:	Presence

1 Scope

The present document is Part 12 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Multimedia Conference Web Service aspects of the interface. All aspects of the Multimedia Conference Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

The Multimedia Conferencing is a simple Web Service that allows the creation of a multimedia conference and the dynamic management of the participants and the media involved.

The underlying model of the service is based on the following entities:

- Conference: a "context" (uniquely identified) to which participants can be added/removed
- Participant: each of the parties involved in the conference. Media can be added/removed for each participant. There may exist a participant that is also the "owner" of the conference, i.e. the user who can end the call and/or be the reference user for billing purposes
- Media: the conference can utilize multiple media streams to support the participants' communication. In particular both audio and video streams are available, including the specific stream direction (i.e. in, out, bidirectional)

An application setting up a multimedia conference must initially invoke the **createConference** method. The result of such invocation is the creation of a "context" that represents a "virtual" room where users can "meet". A unique identifier is assigned to the just-created conference. At this stage no participant is connected yet.

Subsequently the application may wish to add participants to the conference. In order to do so the method **inviteParticipant** can be used. The result of such method is to alert the user of the incoming connection request (eg. the user's terminal rings).

If the application wishes to check whether the user has accepted the invitation (i.e. is connected) it can invoke (at a later time) the **getParticipantInfo** method.

Note that:

- As soon as the first participant connects, the conference becomes "active". The duration of the conference is then measured starting from the moment the conference has became active.
- The initial media set utilized by the participant will depend on the conference type and the media actually supported by the participant's terminal.

During the conference session the application is able to:

- Add (or remove) a specific media stream to a single participant: e.g. adding a video bidirectional stream to a participant that has an audio connection to the conference. This can be obtained by invoking the **addMediaForParticipant** and the **DeleteMediaForParticipant** methods.
- Disconnect a participant from the conference, by invoking the **disconnectParticipant** method
- Retrieve information related to the conference and its status, by invoking **getConferenceInfo** and **getParticipants**

There are different conditions that can determine the end of the conference:

1. The application may invoke the method **endConference**, that "forces" the termination of the conference and the disconnection of all participants.
2. The owner of the conference (if defined) leaves the conference. If the owner is not defined this condition will apply when all the participants have left the conference (disconnected).
3. The conference duration exceeds a maximum value (specified during the conference creation step)

5 Namespaces

The Multimedia Conference interface uses the namespace

www.csapi.org/wsdl/parlayx/multimedia_conference/v2_0

The data types are defined in the namespace

www.csapi.org/schema/parlayx/multimedia_conference/v2_0

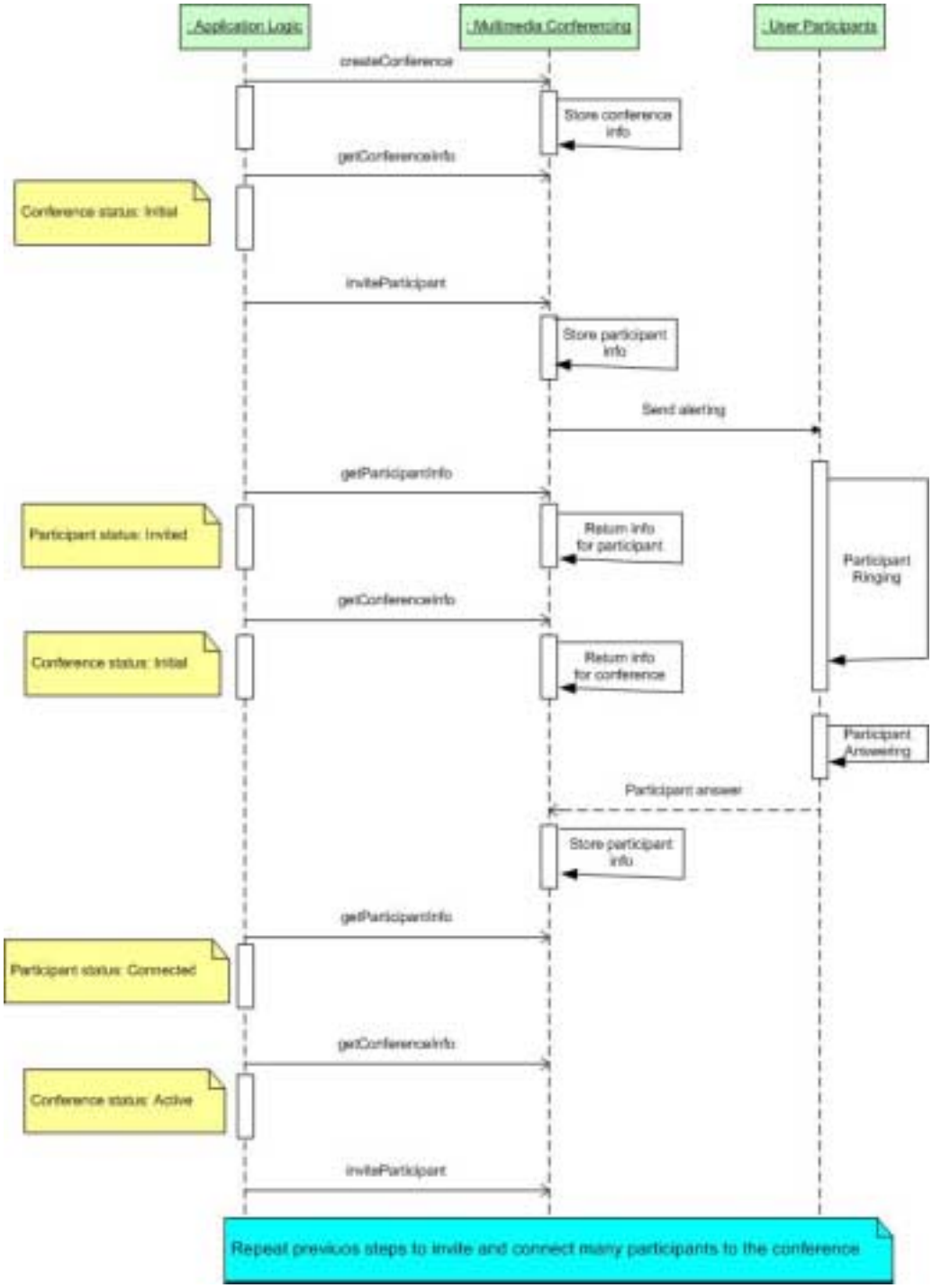
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

The following sequence diagrams illustrate typical scenarios of interaction between an application and the Multimedia Conferencing Web Service.

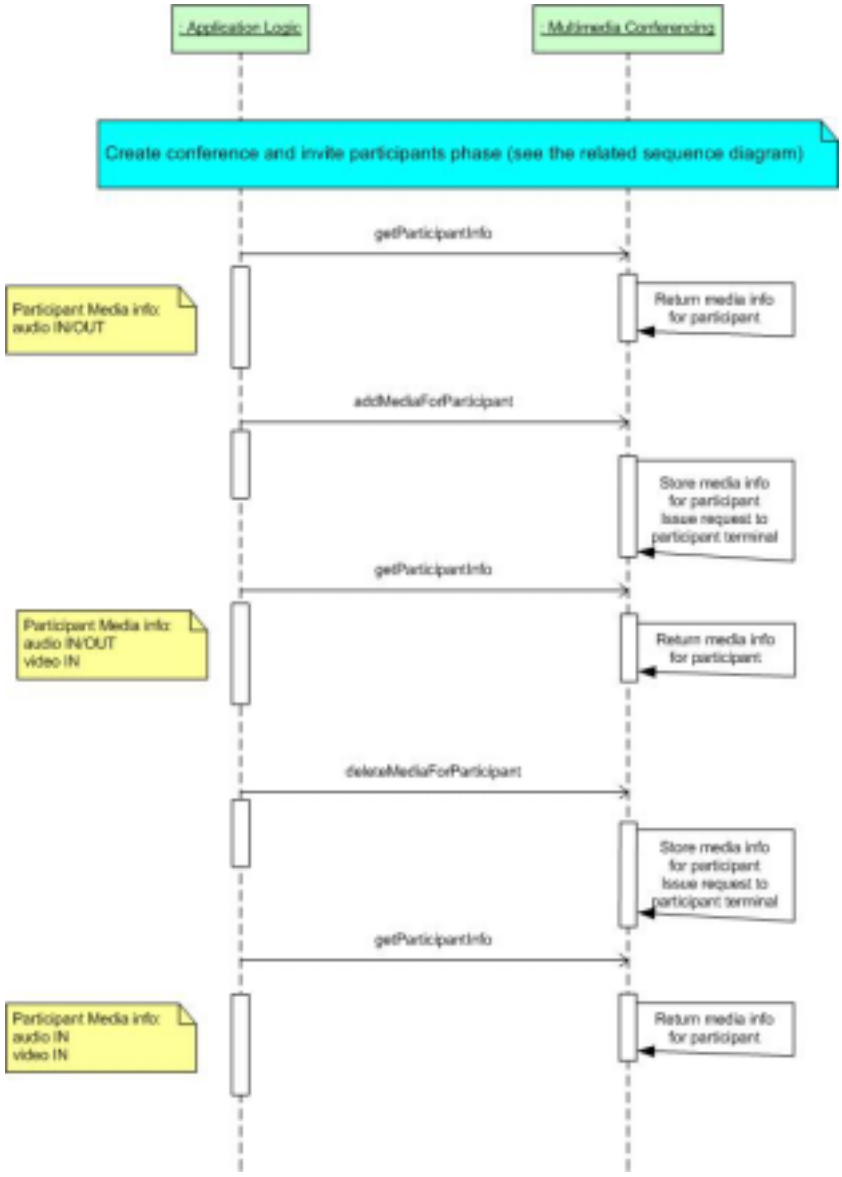
6.1 Setting Up a Conference

Set up a multimedia conference call.



6.2 Adding and Removing Media

On an existing conference call, add media to, or remove media from, a participant.

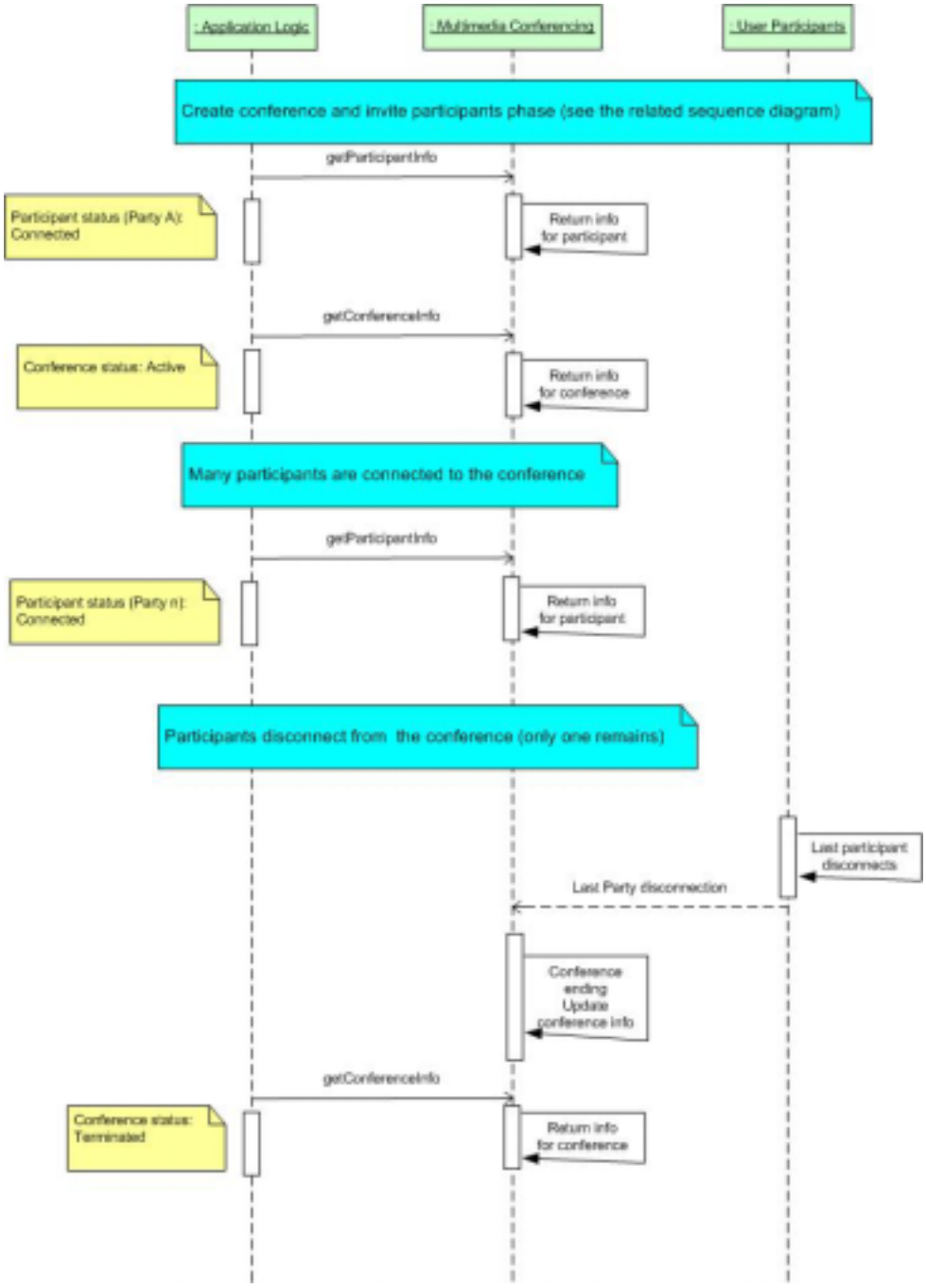


6.3 Conference Owner Disconnects

During a conference call, the conference owner disconnects.

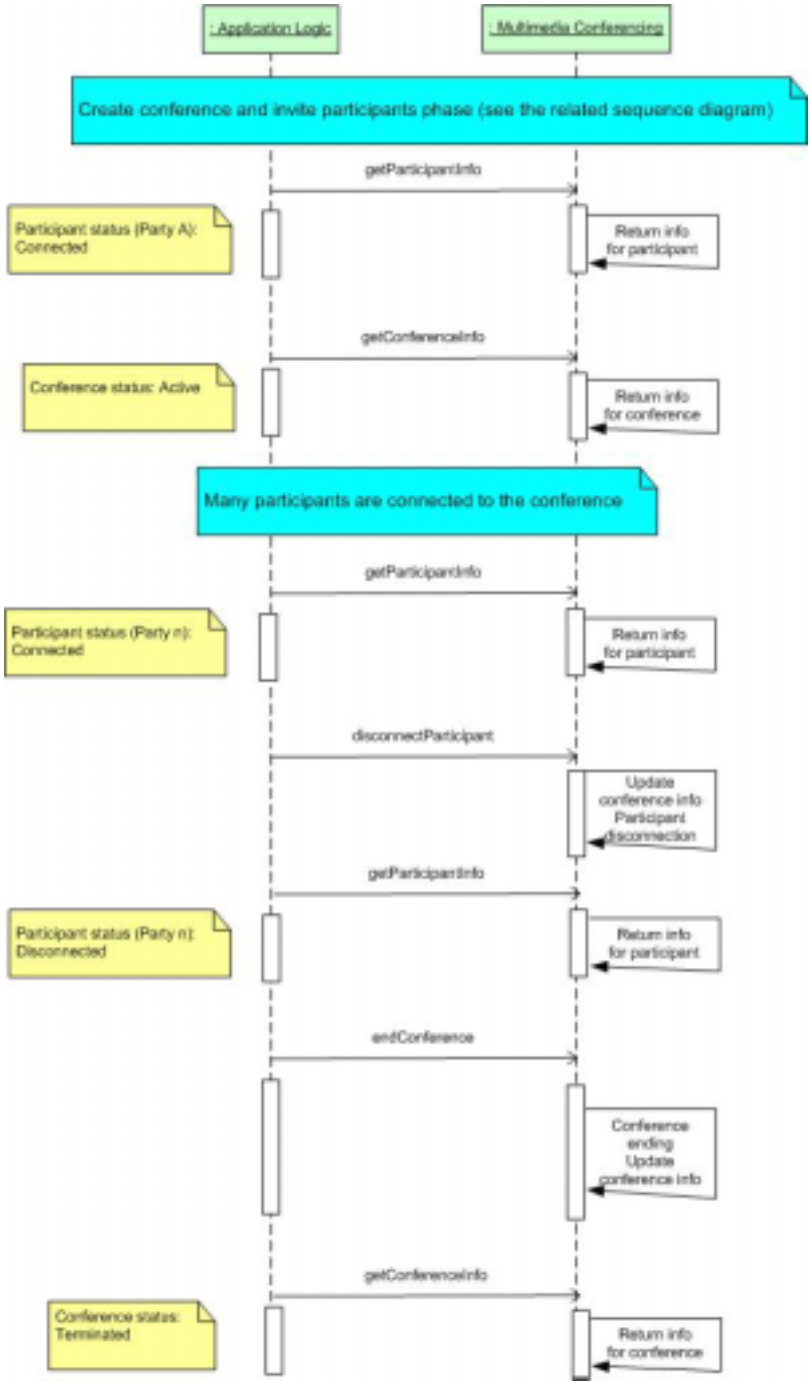
6.4 All Participants Disconnect

End of conference call processing when all participants disconnect.



6.5 Conference Ended by Application

End of conference call processing when the conference is ended by the application.



7 XML Schema Data Type Definition

7.1 ConferenceStatus Enumeration

Element Name	Description
Initial	The conference has been created but no participant is connected yet
Active	The conference is active, i.e. at least one user has connected

Terminated	The conference was terminated

7.2 ConferenceInfo Structure

Name	Type	Description
Status	ConferenceStatus	Status of the conference
StartTime	xsd:dateTime	The time at which the conference was created
Duration	xsd:int	The duration of the conference so far (in seconds)
Owner	xsd:anyURI	Conference owner
NumberOfParticipants	xsd:int	Current number of connected participants
MaximumNumberOfParticipants	xsd:int	Maximum number of participants
ConferenceIdentifier	xsd:string	Conference identifier
ConferenceDescription	xsd:string	Conference description

7.3 ParticipantInfo Structure

Name	Type	Description
Participant	xsd:anyURI	Participant identifier
CodecVideoIn	xsd:string	Codec Video IN
CodecVideoOut	xsd:string	Codec Video OUT
CodecAudioIn	xsd:string	Codec Audio IN
CodecAudioOut	xsd:string	Codec Audio OUT
StartTime	xsd:dateTime	Time this participant joined the conference
Status	ParticipantStatus	Status of participant

7.4 ParticipantStatus Enumeration

Element Name	Description
Invited	Participant invited but not connected yet
Connected	Participant connected
Disconnected	Participant disconnected

7.5 Media Enumeration

Element Name	Description
Audio	Audio media type
Video	Video media type

Chat	Chat media type
Data	Other media type

7.6 MediaDirection Enumeration

Element Name	Description
In	Incoming
Out	Outgoing
InOut	Bidirectional

8 Web Service Interface Definition

8.1 Interface : MultimediaConference

The MultimediaConference interface can be used by an application for creating a multimedia conference call and for dynamically managing the participants and the media involved in the call.

8.1.1 Operation : createConference

The invocation of **createConference** requests to create a multi-media conference with initially no participants connected. The reference to the new multimedia conference is returned in the output parameter.

The conference termination can be driven either by a user action or by the expiring of a maximum duration. In particular, three possible situations are considered. In the first scenario, the concept of the "conference owner" is used. This user that has the control of the call and when the conference owner leaves the conference, all users are disconnected (such a user could be for instance the reference for the conference billing). In this scenario, the optional parameter **conferenceOwner** is present in the method call.

In the second scenario, the conference is terminated when the last participant abandons (in this case the parameter **conferenceOwner** is not present).

A third case is when the optional parameter **maximumDuration** is present: in this situation, when the maximum duration is reached, the conference is terminated.

The selection of the scenario depends on the presence of the optional parameters; if no optional parameter is present, the conference end condition is the disconnection of the last user in conference, if both are present, the conference is terminated when the duration expires (this case could happen if the information concerning the conference owner is needed for billing purposes).

The values **maximumDuration** and **maximumNumberOfParticipants** must not exceed the corresponding service policies otherwise a policy exception is raised.

8.1.1.1 Input message : createConferenceRequest

Part Name	Part Type	Description
ConferenceType	xsd:string	OPTIONAL. Conference type, i.e. one of a list of operator-specific identifiers that indicates how the conference is rendered on the terminals
ConferenceDescription	xsd:string	A text describing the conference

Charging	common:ChargingInformation	OPTIONAL. If present, defines the charge per unit of time consumed on the conference call. If the service does not support charging, a PolicyException (POL0008) will be returned.
MaximumDuration	xsd:int	OPTIONAL. If present it represents the maximum duration of the multimedia conference in seconds. If this parameter is present, it represents the end condition of the conference.
MaximumNumberOfParticipants	xsd:int	Maximum number of participants allowed
ConferenceOwner	xsd:anyURI	OPTIONAL. It is the address of the multimedia conference owner. If this parameter is present, and the maximumDuration is not present, the conference is terminated when this user disconnects, else this information can be used for billing or other purpose

8.1.1.2 Output message : createConferenceResponse

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0008: Charging not supported
- POL0240: Too many participants
- POL0242: Maximum duration exceeded

8.1.2 Operation : getConferenceInfo

The invocation of **getConferenceInfo** requests the information concerning the current status of the multi-media conference call identified by **conferenceIdentifier**.

8.1.2.1 Input message : getConferenceInfoRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier

8.1.2.2 Output message : getConferenceInfoResponse

Part Name	Part Type	Description
ConferenceInfo	ConferenceInfo	Status of the conference

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.3 Operation : endConference

The invocation of **endConference** requests to terminate the multi-media conference call identified by **conferenceIdentifier**.

8.1.3.1 Input message : endConferenceRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier

8.1.3.2 Output message : endConferenceResponse

Part Name	Part Type	Description
None		

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.4 Operation : inviteParticipant

The invocation of **inviteParticipant** requests to add a new participant specified by **participant** to the multi-media conference call identified by **conferenceIdentifier**. The media used for the initial connection of the new participant depends on the conference type and the participant's supported media.

The operation will fail if the conference has already reached the maximum number of participants (as specified in the creation operation).

8.1.4.1 Input message : inviteParticipantRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier
Participant	xsd:anyURI	New participant invited

8.1.4.2 Output message : inviteParticipantResponse

Part Name	Part Type	Description

None		

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0240: Too many participants

8.1.5 Operation : disconnectParticipant

The invocation of **disconnectParticipant** requests to disconnect the participant specified by **participant** from the multi-media conference call identified by **conferenceIdentifier**.

8.1.5.1 Input message : disconnectParticipantRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier
Participant	xsd:anyURI	Participant

8.1.5.2 Output message : disconnectParticipantResponse

Part Name	Part Type	Description
None		

8.1.5.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.6 Operation : getParticipantInfo

The invocation of **getParticipantInfo** requests information concerning the current status of the participant specified by **participant**, in the multi-media conference call identified by **conferenceIdentifier**.

8.1.6.1 Input message : getParticipantInfoRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier
Participant	xsd:anyURI	Participant

8.1.6.2 Output message : getParticipantInfoResponse

Part Name	Part Type	Description
ParticipantInfo	ParticipantInfo	Status of the participant

8.1.6.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.7 Operation : getParticipants

The invocation of **getParticipants** requests information concerning the current status of each participant of the multi-media conference call identified by **conferenceIdentifier**. The output includes participants already disconnected from the conference (if any).

8.1.7.1 Input message : getParticipantsRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier

8.1.7.2 Output message : getParticipantsResponse

Part Name	Part Type	Description
Participants	ParticipantInfo[]	Array containing status information for each participant

8.1.7.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.1.8 Operation : addMediaForParticipant

The invocation of **addMediaForParticipant** requests to add a **media** stream to the media set used by **participant**. The operation is executed on a single participant connected to the multi-media conference call identified by **conferenceIdentifier**. The new media has to be compatible with the type of multimedia conference and the set of media supported by the participant terminal, otherwise the operation will fail.

8.1.8.1 Input message : addMediaForParticipantRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier

Participant	xsd:anyURI	Participant
Media	Media	It identifies the new media stream the participant will receive/send.
MediaDirection	MediaDirection	In indicates the direction of the media stream to add (in, out, etc.)

8.1.8.2 Output message : addMediaForParticipantResponse

Part Name	Part Type	Description
None		

8.1.8.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error
- POL0241: Unavailable media

8.1.9 Operation : deleteMediaForParticipant

The invocation of **deleteMediaForParticipant** requests to remove a **media** stream from the media set used by **participant**. The operation is executed on a single participant connected to the multi-media conference call identified by **conferenceIdentifier**.

8.1.9.1 Input message : deleteMediaForParticipantRequest

Part Name	Part Type	Description
ConferenceIdentifier	xsd:string	Conference identifier
Participant	xsd:anyURI	Participant
Media	Media	It identifies the media the user is not enabled to use any more.
MediaDirection	MediaDirection	In indicates the direction of the media stream to remove (in, out, etc.)

8.1.9.2 Output message : deleteMediaForParticipantResponse

Part Name	Part Type	Description
None		

8.1.9.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

9 Fault Definitions

9.1 PolicyException

9.1.1 POL0240: Too many participants

Too many participants.

Message Id	<POL0240>
Text	Too many participants
Variables	None

9.1.2 POL0241: Unavailable media

Message Id	<POL0241>
Text	Unavailable media
Variables	None

9.1.3 POL0242: Maximum duration exceeded

Message Id	<POL0242>
Text	Maximum duration exceeded. Maximum allowed is %1 seconds.
Variables	%1 – maximum duration set by service policy

10 Service Policies

Name	Type	Description
MaximumDuration	xsd:int	Maximum duration (in seconds) a conference may be set up for
MaximumParticipants	xsd:int	Maximum number of participants a conference may be set up for
ChargingSupported	xsd:boolean	Is charging supported for the createConference operation

Annex A (normative): WSDL for Multimedia Conference

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-12-100-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040360	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	

3GPP TS 29.199-13 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services; Part 13: Address List Management (Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6
2 References 6
3 Definitions and Abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 7
4 Detailed Service Description 7
4.1 Group URI Format 7
4.2 Address List Usage in Services 8
5 Namespaces 8
6 Sequence Diagrams 8
6.1 Manage Groups (Create, Delete, Query, Set Access and Query Access) 8
6.2 Manage Group Members (AddMember, AddMembers, DeleteMember, DeleteMembers, QueryMembers) 9
7 XML Schema Data Type Definition 10
7.1 AccessPermissions Structure 10
7.2 AttributeStatus Enumeration 10
7.3 SimpleAttribute Structure 10
8 Web Service Interface Definition 11
8.1 Interface : GroupManagement 11
8.1.1 Operation : createGroup 11
8.1.1.1 Input message : createGroupRequest 11
8.1.1.2 Output message : createGroupResponse 11
8.1.1.3 Referenced Faults 12
8.1.2 Operation : deleteGroup 12
8.1.2.1 Input message : deleteGroupRequest 12
8.1.2.2 Output message : deleteGroupResponse 12
8.1.2.3 Referenced Faults 12
8.1.3 Operation : queryGroups 12
8.1.3.1 Input message : queryGroupsRequest 13
8.1.3.2 Output message : queryGroupsResponse 13
8.1.3.3 Referenced Faults 13
8.1.4 Operation : setAccess 13
8.1.4.1 Input message : setAccessRequest 13
8.1.4.2 Output message : setAccessResponse 14
8.1.4.3 Referenced Faults 14
8.1.5 Operation : queryAccess 14
8.1.5.1 Input message : queryAccessRequest 14
8.1.5.2 Output message : queryAccessResponse 14
8.1.5.3 Referenced Faults 14
8.2 Interface : Group 14
8.2.1 Operation : addMember 15
8.2.1.1 Input message : addMemberRequest 15
8.2.1.2 Output message : addMemberResponse 15
8.2.1.3 Referenced Faults 15
8.2.2 Operation : addMembers 15
8.2.2.1 Input message : addMembersRequest 15
8.2.2.2 Output message : addMembersResponse 15
8.2.2.3 Referenced Faults 16
8.2.3 Operation: deleteMember 16
8.2.3.1 Input message : deleteMemberRequest 16
8.2.3.2 Output message : deleteMemberResponse 16
8.2.3.3 Referenced Faults 16
8.2.4 Operation: deleteMembers 16

8.2.4.1 Input message : deleteMembersRequest ... 16
8.2.4.2 Output message : deleteMembersResponse ... 17
8.2.4.3 Referenced Faults ... 17
8.2.5 Operation: queryMembers ... 17
8.2.5.1 Input message : queryMembersRequest ... 17
8.2.5.2 Output message : queryMembersResponse ... 17
8.2.5.3 Referenced Faults ... 17
8.2.6 Operation : addGroupAttribute ... 18
8.2.6.1 Input message : addGroupAttributeRequest ... 18
8.2.6.2 Output message : addGroupAttributeResponse ... 18
8.2.6.3 Referenced Faults ... 18
8.2.7 Operation : deleteGroupAttribute ... 18
8.2.7.1 Input message : deleteGroupAttributeRequest ... 18
8.2.7.2 Output message : deleteGroupAttributeResponse ... 18
8.2.7.3 Referenced Faults ... 19
8.2.8 Operation : queryGroupAttributes ... 19
8.2.8.1 Input message : queryGroupAttributesRequest ... 19
8.2.8.2 Output message : queryGroupAttributesResponse ... 19
8.2.8.3 Referenced Faults ... 19
8.2.9 Operation : addGroupMemberAttribute ... 19
8.2.9.1 Input message : addGroupMemberAttributeRequest ... 19
8.2.9.2 Output message : addGroupMemberAttributeResponse ... 19
8.2.9.3 Referenced Faults ... 20
8.2.10 Operation : deleteGroupMemberAttribute ... 20
8.2.10.1 Input message : deleteGroupMemberAttributeRequest ... 20
8.2.10.2 Output message : deleteGroupMemberAttributeResponse ... 20
8.2.10.3 Referenced Faults ... 20
8.2.11 Operation : queryGroupMemberAttributes ... 20
8.2.11.1 Input message : queryGroupMemberAttributesRequest ... 20
8.2.11.2 Output message : queryGroupMemberAttributesResponse ... 21
8.2.11.3 Referenced Faults ... 21
8.3 Interface : Member ... 21
8.3.1 Operation : addMemberAttribute ... 21
8.3.1.1 Input message :addMemberAttributeRequest ... 21
8.3.1.2 Output message : addMemberAttributeResponse ... 21
8.3.1.3 Referenced Faults ... 21
8.3.2 Operation : queryMemberAttributes ... 21
8.3.2.1 Input message :queryMemberAttributesRequest ... 22
8.3.2.2 Output message : queryMemberAttributesResponse ... 22
8.3.2.3 Referenced Faults ... 22
8.3.3 Operation : deleteMemberAttribute ... 22
8.3.3.1 Input message :deleteMemberAttributeRequest ... 22
8.3.3.2 Output message : deleteMemberAttributeResponse ... 22
8.3.3.3 Referenced Faults ... 22
9 Fault Definitions ... 23
9.1 Fault : PolicyException ... 23
9.1.1 POL0210: Too many members in group ... 23
9.1.2 Subgroups not supported ... 23
9.1.3 Group name too long ... 23
9.1.4 Group already exists ... 23
10 Service Policies ... 23
Annex A (normative): WSDL for Address List Management ... 25
Annex B (informative): Change history ... 26

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 13 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	**Address List Management**
Part 14:	Presence

1 Scope

The present document is Part 13 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Address List Management Web Service aspects of the interface. All aspects of the Address List Management Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service Policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and The Parlay Group.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

3 Definitions and Abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.199-1 [6] and the following apply.

Group: A group is a container for a set of addresses, it is not an address itself. When a group contain one or more groups, logically the group contains the set of addresses it holds, plus the set of addresses that any contained group holds (including any addresses contained in groups that a contained group holds).

Group Resolution: When a group is processed by a service, it expands the group (and any nested groups) into a set of addresses. The resulting set of addresses contains no groups, and any duplicate addresses are removed. Thus, a resolved group may be considered an exclusive union of all of its contained members.

Application managed group: A group created and managed outside of the network, requiring the group members to be passed into the network for processing.

Network managed group: A group created and managed within a network, allowing Web Services to reference the members of a group using the group name.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] apply.

4 Detailed Service Description

This specification defines two related interfaces, one to manage the groups themselves – creation, deletion, query and access right management. The second interface manages the members within a group, supporting add, delete and query operations.

Addresses are not created using this service, they must already exist.

4.1 Group URI Format

A group URI is consistent with the style defined in [RFC 2396], supporting the following URI style which is used in schemes such as sip and mailto,

scheme:dept1294@mydivision.mycompany.serviceprovider.com

The group URI consists of the following discrete elements,

Scheme: selected by the provider of the group URI

Group name: following the conventions of [RFC 2396]

Suffix: may be added by Service Provider (if allowed by creation operation) to create a unique name when the Prefix + Group name already exists.

Sub-domain: defined by the requester, this is contained within the domain provided by the service provider.

Domain: defined by the Service Provider, and cannot be specified by the application.

This definition of a group URI enables flexibility on the part of the Service Provider and the Requester, while ensuring unique groups are created and providing transparency of implementation of group storage.

The following are some group URI examples.

- sip:salesteam@sales.acme.anytelco.com
- sip:salesteam1@sales.acme.anytelco.com
- mailto:fieldservice@cityofaustin.anytelco.com
- group:mailroom@bldg001.acme.anytelco.com

These examples show (1)(2) use of prefix to create unique names, (1)(3) use of different defined schemes, and (4) use of a service provider defined scheme.

4.2 Address List Usage in Services

When a service has a requirement to support groups of address lists, it may satisfy this requirement by utilizing network managed groups. The group URI is passed to the service, and this group URI is resolved to the set of URIs contained within the group. If one or more group URIs are provided in a set of URIs to a service, the service will replace each group URI with its set of contained URIs, and the service processing will apply to the unique union of URIs generated.

If supported by the service policy, zero or more of the set of URIs contained within a group may be themselves group URIs, which would also be resolved. Thus, in this case, the list of URIs that the service would process would be the union of individual URIs (as a set with no duplicates).

Unless specifically defined in the semantics of a service, the expected semantic for the results of a service operation will be presented as the results for the set of URIs as processed (the union of non-group and group provided URIs), without group URIs included in the result. This eliminates a variety of complexity issues including duplicate URIs in multiple groups and the differences between a group URI and a URI referring to an endpoint.

5 Namespaces

The GroupManagement interface uses the namespace

www.csapi.org/wsdl/parlayx/group_management/v2_0

The Group interface uses the namespace

www.csapi.org/wsdl/parlayx/group/v2_0

The GroupMember interface uses the namespace

www.csapi.org/wsdl/parlayx/group_member/v2_0

The data types are defined in the namespace

www.csapi.org/schema/parlayx/group/v2_0

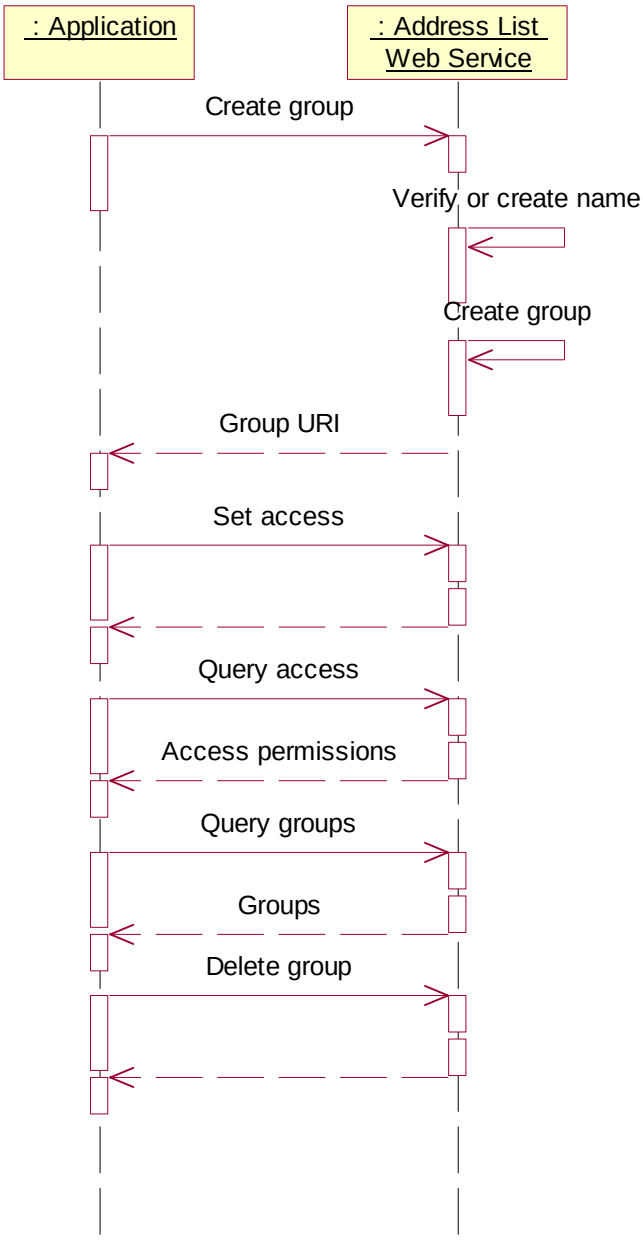
The ‘xsd’ namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name ‘xsd’ is not semantically significant.

6 Sequence Diagrams

6.1 Manage Groups (Create, Delete, Query, Set Access and Query Access)

Pattern: Request / Response

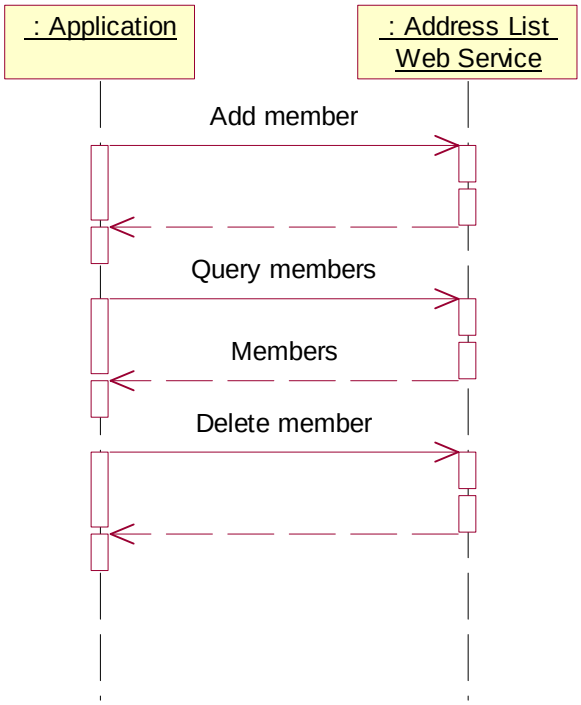
The group management functions are shown in this diagram, showing a sequence including the creation of a group, setting access permissions to the group, querying those permissions, query of groups and finally deletion of a group.



6.2 Manage Group Members (AddMember, AddMembers, DeleteMember, DeleteMembers, QueryMembers)

Pattern: Request / Response

The group membership functions are shown in this diagram, showing the two add, two delete, and the query function.



7 XML Schema Data Type Definition

7.1 AccessPermissions Structure

List of access permissions that may be assigned to a requester associated with a group.

Name	Type	Description
AdminPermission	xsd:boolean	Requester has admin permission for the group
AddPermission	xsd:boolean	Requester can add members to a group
DeletePermission	xsd:boolean	Requester can delete members from a group
QueryPermission	xsd:boolean	Requester can query members in a group

7.2 AttributeStatus Enumeration

Enumeration	Description
Valid	Attribute is valid
Unknown	Attribute is not defined
Denied	Access to the attribute is denied

7.3 SimpleAttribute Structure

Attribute representing a name and an associated value.

Name	Type	Description

Name	xsd:string	Name of the attribute
Type	xsd:string	Type of the attribute. The value is always a string, but this provides information on the format of the value.
Value	xsd:string	Value of the attribute
Status	AttributeStatus	Status of the attribute

8 Web Service Interface Definition

The Address List Management service consists of three interfaces,

- GroupManagement which manages creation and access to groups that hold the address lists
- Group which manages the content of the address list
- GroupMember which represents an address list entry and its associated properties

Together these provide the interfaces to create and manage address lists, enabling these groups to be used by other services through this common capability.

8.1 Interface : GroupManagement

The GroupManagement interface provides the administration interface for creating, deleting, querying and managing access rights for groups. The format of the group name is specified in the Detailed Service Description (see section **Error! Reference source not found.**).

8.1.1 Operation : createGroup

Create a new group. The requester provides the name for the group and the domain segment in which the group is to be stored. A domain segment is used, since the full domain will consist of the domain segment provided by the requester (e.g. ‘sales.mycompany’) plus a period separator (‘.’) per [RFC 2396] and the domain segment provided by the Service Provider (e.g. ‘serviceprovider.com’).

To avoid name conflicts, since group URIs must be unique, an automatic naming capability is provided which will append a suffix to the name provided if the name is already used within the domain. If the AutoName is set to ‘true’ and the fully qualified name is not unique, then the name will have a suffix added and the unique name will be provided in the result. For example, if the group ‘sales@mycompany.serviceprovider.com’ was already defined, a suffix would be added and the result could be ‘sales1@mycompany.serviceprovider.com’. If the AutoName is set to ‘false’, then a PolicyException is thrown if the group URI is not unique.

8.1.1.1 Input message : createGroupRequest

Part Name	Part Type	Description
Name	xsd:string	Name of group to be included in group name
Domain	xsd:string	Domain segment to be contained within the domain provided by the Service Provider. May be hierarchial using period separators (see [RFC 2396])
AutoName	xsd:boolean	If false, name must be unique or it will not be created. If true, a suffix will be added to the name if it is not unique.

8.1.1.2 Output message : createGroupResponse

Part Name	Part Type	Description

Result	xsd:anyURI	Fully qualified group name

8.1.1.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error
- POL0212: Group name too long
- POL0213 : Group already exists

8.1.2 Operation : deleteGroup

Delete a group.

8.1.2.1 Input message : deleteGroupRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Name of group to delete

8.1.2.2 Output message : deleteGroupResponse

Part Name	Part Type	Description
None		

8.1.2.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.1.3 Operation : queryGroups

Group information can be retrieved from the network, with two types of search, one that retrieves groups only from a single sub-domain and one that returns groups from the sub-domain and its sub-domains.

An example demonstrates the two search types. The following example data is used.

Dept123@region1.sales.mycompany.serviceprovider.com

Dept245@region2.sales.mycompany.serviceprovider.com

Dept348@sales.mycompany.serviceprovider.com

Dept367@sales.mycompany.serviceprovider.com

Dept875@finance.mycompany.serviceprovider.com

For a search using the search domain ‘sales.mycompany’, with the hierarchy set to ‘false’, the result will contain

Dept348@sales.mycompany.serviceprovider.com

Dept367@sales.mycompany.serviceprovider.com

If the same search domain ‘sales.mycompany’ is used, but the hierarchy set to ‘true’, the result will contain,

Dept123@region1.sales.mycompany.serviceprovider.com

Dept245@region2.sales.mycompany.serviceprovider.com

Dept348@sales.mycompany.serviceprovider.com

Dept367@sales.mycompany.serviceprovider.com

8.1.3.1 Input message : queryGroupsRequest

Part Name	Part Type	Description
SearchDomain	xsd:string	Sub-domain to retrieve groups from
Hierarchy	xsd:boolean	Follow hierarchy under search name.

8.1.3.2 Output message : queryGroupsResponse

Part Name	Part Type	Description
Result	xsd:anyURI [0..unbounded]	Array of items matching search criteria.

8.1.3.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.1.4 Operation : setAccess

Access to manage the elements within a group may be provided independently from the access to manage the group itself. This operation enables the group administrator to specify the requester and the operations the requester is permitted to perform through the Group interface.

The access rights are absolute, if a requester has ‘query’ access currently and ‘add’ access is to be added, then the request requires both ‘add’ and ‘query’ rights to be set to ‘true’. Likewise, any right that is set to ‘false’ will be revoked.

8.1.4.1 Input message : setAccessRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to grant access to.
Requester	xsd:string	Requester to grant access to.
AdminPermission	xsd:Boolean	Permission to manage group

AddPermission	xsd:Boolean	Permission to add members to the group.
DeletePermission	xsd:Boolean	Permission to delete members from the group.
QueryPermission	xsd:Boolean	Permission to query members in the group.

8.1.4.2 Output message : setAccessResponse

Part Name	Part Type	Description
None.		

8.1.4.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.1.5 Operation : queryAccess

Query the access permissions for a requester on a group.

8.1.5.1 Input message : queryAccessRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to which permissions are to be granted.
Requester	xsd:string	Requester to retrieve access permissions for.

8.1.5.2 Output message : queryAccessResponse

Part Name	Part Type	Description
Permissions	AccessPermissions	List of permissions that a requester has.

8.1.5.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2 Interface : Group

The Group interface provides the administration interface for creating, deleting, querying members within a group.

8.2.1 Operation : addMember

Add a member to a group. If the new member is a group, and if nested group support is provided, this will add the group URI as a reference to the list of members (it will not expand the contents of the group within this group). A group may not be added recursively, an attempt to do so will result in a ServiceException.

To add a group as a member of a group, the requester must have query permission on the group to be added.

8.2.1.1 Input message : addMemberRequest

Part Name	Part Type	Description
Group	xsd:anyURI	URI of group to which a member is to be added.
Member	xsd:anyURI	Member to add to the group.

8.2.1.2 Output message : addMemberResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error
- POL0210: Too many members in group
- POL0211: Subgroups not allowed

8.2.2 Operation : addMembers

Add an array of members to a group. If nested group support is provided, this will add any group URIs, as references, to the list of members (it will not expand the contents of any groups within this group). No group may be added recursively, an attempt to do so will result in a ServiceException, and none of the members will be added to the group.

To add a group as a member of a group, the requester must have query permission on the group to be added.

8.2.2.1 Input message : addMembersRequest

Part Name	Part Type	Description
Group	xsd:anyURI	URI of group to which a member is added.
Members	xsd:anyURI [0..unbounded]	Member to add to the group.

8.2.2.2 Output message : addMembersResponse

Part Name	Part Type	Description
None		

8.2.2.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error
- POL0210: Too many members in group
- POL0211: Subgroups not allowed

8.2.3 Operation: deleteMember

Delete a member from a group. The member may only be removed from this group. If nested groups are supported, the member will not be removed from any nested group. Removal of a group URI will remove that group URI reference from this group, is will not delete the group.

8.2.3.1 Input message : deleteMemberRequest

Part Name	Part Type	Description
Group	xsd:anyURI	URI of group.
Member	xsd:anyURI	Member to delete from the group.

8.2.3.2 Output message : deleteMemberResponse

Part Name	Part Type	Description
None		

8.2.3.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.4 Operation: deleteMembers

Delete an array of members from a group. The members may only be removed from this group. If nested groups are supported, the members will not be removed from any nested group. Removal of a group URI will remove that group URI reference from this group, is will not delete the group. If the array contains URIs that are not in the group, they will be ignored and no fault will be generated.

8.2.4.1 Input message : deleteMembersRequest

Part Name	Part Type	Description
Group	xsd:anyURI	URI of group.

Members	xsd:anyURI [0..unbounded]	Member to delete from the group.

8.2.4.2 Output message : deleteMembersResponse

Part Name	Part Type	Description
None		

8.2.4.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.5 Operation: queryMembers

Get the list of members contained within a group.

If nested groups are supported, then the member list may contain group URIs as members. Therefore, two manners are supported for retrieving the list of members – with members resolved and without.

- If ResolveGroups is 'true', then the exclusive union of all the members contained within the group, and any nested subgroups, is the result (exclusive union means that after retrieving all members, duplicate members are removed).
- If ResolveGroup is 'false', then the group members are returned including group URIs as members of the group. If members within nested groups are required, subsequent calls to this operation with those groups may be used to retrieve those members.

If nested groups are not supported, the value of ResolveGroups is ignored.

8.2.5.1 Input message : queryMembersRequest

Part Name	Part Type	Description
Group	xsd:anyURI	URI of group.
ResolveGroups	xsd:boolean	If true, return set of members after resolving groups (including subgroups). If false, return members including group references.

8.2.5.2 Output message : queryMembersResponse

Part Name	Part Type	Description
Members	xsd:anyURI [0..unbounded]	Members of group.

8.2.5.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error

- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.6 Operation : addGroupAttribute

Groups may have attributes associated with the group. To avoid conflicts, attribute names that start with Group are reserved for use as defined within this specification,

- Group.Description
- Group.ExpiryDate

Attributes may be added or updated by those with admin or add permission on the specified group.

8.2.6.1 Input message : addGroupAttributeRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to set attribute for
Value	SimpleAttribute	Attribute to add, or update

8.2.6.2 Output message : addGroupAttributeResponse

Part Name	Part Type	Description
None		

8.2.6.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.7 Operation : deleteGroupAttribute

Groups may have attributes removed by those with admin or delete permission on the specified group.

8.2.7.1 Input message : deleteGroupAttributeRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to set attribute for
AttributeName	xsd:string	Name of attribute to delete

8.2.7.2 Output message : deleteGroupAttributeResponse

Part Name	Part Type	Description
None		

8.2.7.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.8 Operation : queryGroupAttributes

Query the attributes for a group by those with admin or read permission on the specified group.

8.2.8.1 Input message : queryGroupAttributesRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to get attributes for.

8.2.8.2 Output message : queryGroupAttributesResponse

Part Name	Part Type	Description
Result	SimpleAttribute [0..unbounded]	Group attributes.

8.2.8.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.9 Operation : addGroupMemberAttribute

Group members may have attributes that are within the context of a group in which they belong.

Group member attributes may be added or updated by those with admin or add permission on the specified group.

8.2.9.1 Input message : addGroupMemberAttributeRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to set attribute for.
Member	xsd:anyURI	Member to set attribute for
Value	SimpleAttribute	Attribute to add, or update

8.2.9.2 Output message : addGroupMemberAttributeResponse

Part Name	Part Type	Description

None		

8.2.9.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.10 Operation : deleteGroupMemberAttribute

Group members may have attributes removed by those with admin or delete permission on the specified group.

8.2.10.1 Input message : deleteGroupMemberAttributeRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to delete attribute from
Member	xsd:anyURI	Member to delete attribute from
AttributeName	xsd:string	Name of attribute to remove

8.2.10.2 Output message : deleteGroupMemberAttributeResponse

Part Name	Part Type	Description
None		

8.2.10.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.2.11 Operation : queryGroupMemberAttributes

Query the attributes for a group member by those with admin or read permission on the specified group.

8.2.11.1 Input message : queryGroupMemberAttributesRequest

Part Name	Part Type	Description
Group	xsd:anyURI	Group to get attributes for.
Member	xsd:anyURI	Member to set attribute for

8.2.11.2 Output message : queryGroupMemberAttributesResponse

Part Name	Part Type	Description
Result	SimpleAttribute [0..unbounded]	Group attributes.

8.2.11.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.3 Interface : Member

The Member interface provides access to information related to a particular entity.

8.3.1 Operation : addMemberAttribute

Add member attribute. If an attribute with this name exists, its value will be replaced with the value provided in this operation.

8.3.1.1 Input message :addMemberAttributeRequest

Part Name	Part Type	Description
Member	xsd:anyURI	Member to add attribute to
Data	SimpleAttribute	Attribute to add to member

8.3.1.2 Output message : addMemberAttributeResponse

Part Name	Part Type	Description
None		

8.3.1.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.3.2 Operation : queryMemberAttributes

Query attributes of a member. If any attributes requested do not exist, they will not be included in the result.

8.3.2.1 Input message :queryMemberAttributesRequest

Part Name	Part Type	Description
Member	xsd:anyURI	Member to query attributes for
AttributeNames	xsd:string [0..unbounded]	List of attribute names to retrieve

8.3.2.2 Output message : queryMemberAttributesResponse

Part Name	Part Type	Description
Result	SimpleAttribute [0..unbounded]	List of attributes

8.3.2.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

8.3.3 Operation : deleteMemberAttribute

Delete attribute from a member. If the attribute specified does not exist, it will be ignored.

8.3.3.1 Input message :deleteMemberAttributeRequest

Part Name	Part Type	Description
Member	xsd:anyURI	Member to remove attributes from
AttributeName	xsd:string	List of attribute names to delete

8.3.3.2 Output message : deleteMemberAttributeResponse

Part Name	Part Type	Description
None		

8.3.3.3 Referenced Faults

ServiceException from [COMMON]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [COMMON]

- POL0001: Policy error

9 Fault Definitions

9.1 Fault : PolicyException

9.1.1 POL0210: Too many members in group

Number of members in a group exceeds the number allowed by the Service Policy (MaxGroupMembers).

Message Id	<POL0210>
Text	Attempt to exceed maximum number of members in a group. Maximum number allowed is %1.
Variables	%1 = Maximum number allowed by Service Policy.

9.1.2 Subgroups not supported

Attempt to add a subgroup not permitted by Service Policy (SupportNestedGroups).

Message Id	<POL0211>
Text	Attempted to add a group to an existing group. Subgroups are not supported.
Variables	None.

9.1.3 Group name too long

Length of group name exceeds the length allowed by the Service Policy (MaxGroupLength)

Message Id	<POL0212>
Text	Group name is too long. Maximum length allowed is %1.
Variables	%1 = Maximum length allowed by Service Policy.

9.1.4 Group already exists

If the group name is not unique and the AutoName is set to ‘false’, then a PolicyException is returned since the group name already exists.

Message Id	POL0213
Text	Group URI %1 already exists. Group not created.
Variables	%1 = Group URI

10 Service Policies

Name	Type	Description
MaxGroupLength	xsd:int	Maximum length of the group name (user portion)
MaxGroupMembers	xsd:int	Maximum number of members in a group
SupportNestedGroups	xsd:boolean	Can a group member be a group URI

Annex A (normative): WSDL for Address List Management

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-13-100-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040360	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	

3GPP TS 29.199-14 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Core Network;
Open Service Access (OSA);
Parlay X Web Services;
Part 14: Presence
(Release 6)

Keywords
API, OSA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

1 Scope 6

2 References 6

3 Definitions and Abbreviations 7
3.1 Definitions 7
3.2 Abbreviations 7

4 Detailed Service Description 8

5 Namespaces 9

6 Sequence Diagrams 9
6.1 Interface Flow Overview 9

7 XML Schema Data Type Definition 10
7.1 PresenceAttributeType Enumeration 10
7.2 ActivityValue Enumeration 11
7.3 PlaceValue Enumeration 11
7.4 PrivacyValue Enumeration 12
7.5 SphereValue Enumeration 12
7.6 CommunicationMeansType Enumeration 13
7.7 CommunicationMeans Structure 13
7.8 CommunicationValue Structure 13
7.9 OtherValue Structure 13
7.10 PresenceAttribute Structure 14
7.11 SubscriptionRequest Structure 14
7.12 PresencePermission Structure 14

8 Web Service Interface Definition 14
8.1 Interface : PresenceConsumer 15
8.1.1 Operation : subscribePresence 15
8.1.1.1 Input message : subscribePresenceRequest 15
8.1.1.2 Output message : subscribePresenceResponse 15
8.1.1.3 Referenced Faults 15
8.1.2 Operation : getUserPresence 16
8.1.2.1 Input message : getUserPresenceRequest 16
8.1.2.2 Output message : getUserPresenceResponse 16
8.1.2.3 Referenced Faults 16
8.1.3 Operation : startPresenceNotification 16
8.1.3.1 Input message : startPresenceNotificationRequest 17
8.1.3.2 Output message : startPresenceNotificationResponse 17
8.1.3.3 Referenced Faults 17
8.1.4 Operation : endPresenceNotification 18
8.1.4.1 Input message : endPresenceNotificationsRequest 18
8.1.4.2 Output message : endPresenceNotificationResponse 18
8.1.4.3 Referenced Faults 18
8.2 Interface : PresenceNotification 18
8.2.1 Operation : statusChanged 18
8.2.1.1 Input message : statusChangedRequest 18
8.2.1.2 Output message : statusChangedResponse 18
8.2.1.3 Referenced Faults 18
8.2.2 Operation : statusEnd 19
8.2.2.1 Input message : statusEndRequest 19
8.2.2.2 Output message : statusEndResponse 19
8.2.2.3 Referenced Faults 19
8.2.3 Operation : notifySubscription 19
8.2.3.1 Input message : notifySubscriptionRequestt 19
8.2.3.2 Output message : notifySubscriptionResponse 19
8.2.4 Operation : subscriptionEnded 19

8.2.4.1 Input message : subscriptionEndedRequest 19
8.2.4.2 Output message : subscriptionEndedResponse 20
8.3 Interface : PresenceSupplier 20
8.3.1 Operation : publish 20
8.3.1.1 Input message : publishRequest 20
8.3.1.2 Output message : publishResponse 20
8.3.1.3 Referenced Faults 20
8.3.2 Operation : getOpenSubscriptions 20
8.3.2.1 Input message : getOpenSubscriptionsRequest 20
8.3.2.2 Output message : getOpenSubscriptionsResponse 20
8.3.2.3 Referenced Faults 21
8.3.3 Operation : updateSubscriptionAuthorization 21
8.3.3.1 Input message : updateSubscriptionAuthorizationRequest 21
8.3.3.2 Output message updateSubscriptionAuthorizationResponse 21
8.3.3.3 Referenced Faults 21
8.3.4 Operation : getMyWatchers 21
8.3.4.1 Input message : getMyWatchersRequest 21
8.3.4.2 Output message : getMyWatchersResponse 22
8.3.4.3 Referenced Faults 22
8.3.5 Operation : getSubscribedAttributes 22
8.3.5.1 Input message : getSubscribedAttributesRequest 22
8.3.5.2 Output message : getSubscribedAttributesResponse 22
8.3.5.3 Referenced Faults 22
8.3.6 Operation : blockSubscription 22
8.3.6.1 Input message : blockSubscriptionRequest 22
8.3.6.2 Output message : blockSubscriptionResponse 23
8.3.6.3 Referenced Faults 23

9 Fault Definitions 23
9.1 ServiceException 23
9.1.1 SVC0220 : No subscription request 23
9.1.2 SVC0221 : Not a watcher 23

10 Service Policies 23

Annex A (normative): WSDL of Presence API 25

Annex B (informative): Change history 26

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

3GPP acknowledges the contribution of the Parlay X Web Services specifications from The Parlay Group. The Parlay Group is pleased to see 3GPP acknowledge and publish this specification, and the Parlay Group looks forward to working with the 3GPP community to improve future versions of this specification.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 14 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Parlay X Web Services, as identified below. The **Parlay X Web Services specification** (3GPP TS 29.199) is structured in the following Parts:

Part 1:	Common
Part 2:	Third Party Call
Part 3:	Call Notification
Part 4:	Short Messaging
Part 5:	Multimedia Messaging
Part 6:	Payment
Part 7:	Account Management
Part 8:	Terminal Status
Part 9:	Terminal Location
Part 10:	Call Handling
Part 11:	Audio Call
Part 12:	Multimedia Conference
Part 13:	Address List Management
Part 14:	**Presence**

1 Scope

The present document is Part 14 of the Stage 3 Parlay X Web Services specification 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 concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Presence Web Service aspects of the interface. All aspects of the Presence Web Service are defined here, these being:

- Name spaces
- Sequence Diagrams
- Data definitions
- Interface specification plus detailed method descriptions
- Fault definitions
- Service policies
- WSDL Description of the interfaces

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI TISPAN and the Parlay Consortium.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA); Stage 1".

[3] 3GPP TS 23.127: "Virtual Home Environment (VHE) / Open Service Access (OSA)".

[4] 3GPP TS 22.101: "Service aspects; Service principles".

[5] XML Schema, available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] 3GPP TS 29.199-1: "Open Service Access (OSA); Parlay X web services; Part 1: Common".

[7] Event notification filtering: draft-ietf-simple-event-filter-funct-01.txt

[8] 3GPP TS 29.198-14: "Open Service Access (OSA) Application Programming Interface (API); Part 14: Presence and Availability Management (PAM)".

[9] SIP SIMPLE (draft-ietf-simple-presence-10.txt)

[10] XMPP (Jabber)

[11] Rich Presence Information Data Format draft-ietf-simple-rpid-04.txt

[12] 3GPP TS 23.141: "Presence service; Architecture and functional description; Stage 2".

[13] Parlay X group management

[14] SIP-Specific Event Notification: RFC3265

[15] 3GPP XCAP Protocol - draft-ietf-simple-xcap-02.txt

3 Definitions and Abbreviations

3.1 Definitions

In addition to the terms and definitions given in TS 29.199-1 [6] the following definitions apply.

We use the names *watcher* and *presentity* to denote the role of the client connected to the presence services. Like in OSA/Parlay PAM [8] the watcher and the presentity have to be associated to identities registered to the system, i.e. users, groups of users or organizations.

An *Identity* represents a user in the real world. See OSA/Parlay PAM identities [8, section 4.4.1].

Presence information: Presence information consists of a set of attributes that characterize the presentity such as current activity, environment, communication means and contact addresses. Only the system and the presentity have direct access to this information, which may be collected and aggregated from **several** devices associated to the presentity.

Applications: for Instant Messaging, Push to Talk, or call control and other purposes may become clients of the presence web service. We assume that these applications belong to a watcher and authenticate to the services in the name of the watcher.

Presence attributes: contain information about a presentity. An attribute has a name and a value and can be supplied by any device, application or network module that can be associated to the presentity's identity. A watcher can obtains attributes only after he has successfully subscribed to them. Examples for attributes are activity, location type, communication means, etc.

Subscription: Before a watcher can access presence data, he has to subscribe to it. One possibility the API provides is an end-to-end subscription concept, in which only identities that have accepted a subscription to their presence can be addressed. Subscriptions can be also automatically handled by server policies edited by the presentity or other authorized users. The service/protocol to manage those policies is out of the scope of this specification.
Note: This definition is not related to the term “subscription” in [1].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.199-1 [6] and the following apply.

ACL	Access Control List
DMS	Data Manipulation Server
GM	Group Management
IETF	Internet Engineering Taskforce
IMS	IP Multimedia Subsystem
ISC	IP multimedia subsystem Service Control interface
MMS	Multimedia Message Service
PAM	Presence and Availability Management
RLS	Resource List Server
SCF	Service Capability Feature
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SMS	Short Message Service

URI	Uniform Resource Identifier
WS	Web Service
WSDL	Web Services Definition Language
XCAP	Extensible Markup Language (XML) Configuration Access Protocol
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XSD	XML Schema Definition

4 Detailed Service Description

The presence service allows for presence information to be obtained about one or more users and to register presence for the same. It is assumed that the typical client of these interfaces is either a supplier or a consumer of the presence information. An Instant Messaging application is a canonical example of such a client of this interface.

Figure 4-1 shows the architecture of the presence web service and the underlying services. The OSA/Parlay PAM SCF is the straightforward option and implements the presence server with extended identity-, device capability,- and presence agent management. OSA/Parlay PAM allows aggregation of presence information from internet, mobile and enterprise users, etc. using a presence transport network of SIP or XMPP servers. The Presence Web Service can however communicate directly for example with IMS presence network elements (presence and resource list servers) using the ISC (SIP/SIMPLE) protocol interface.

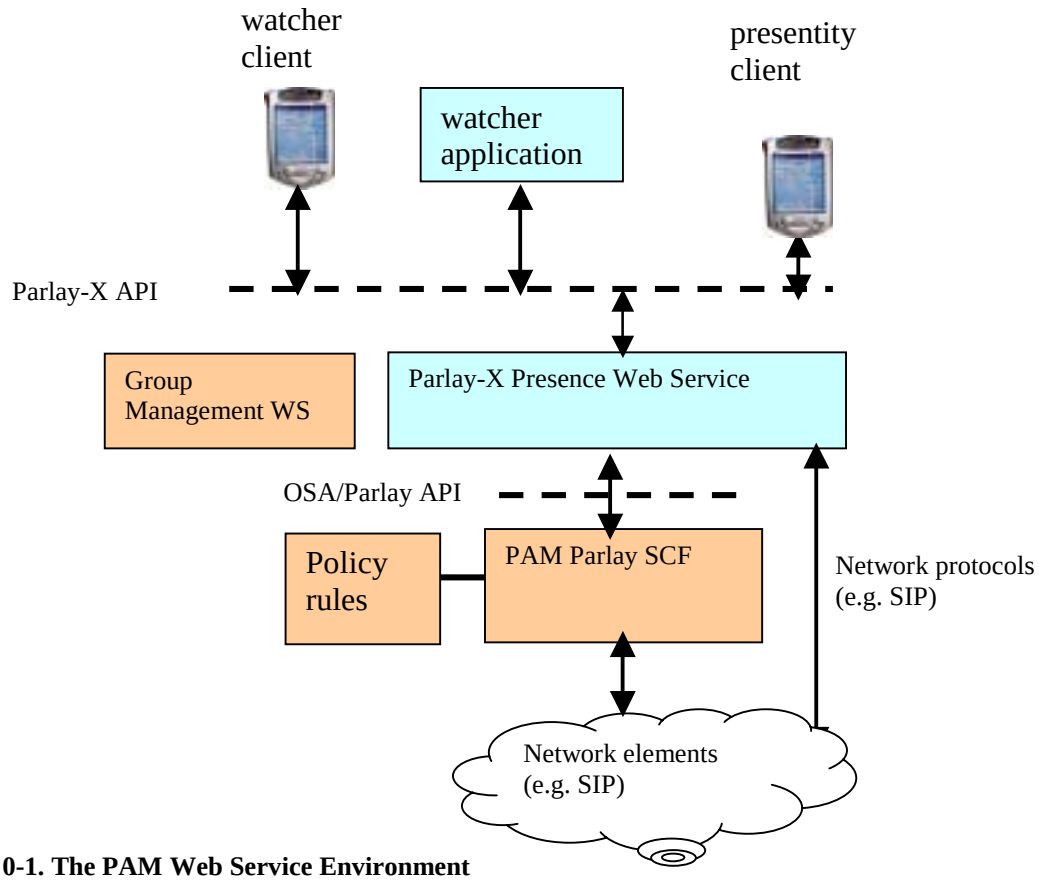


Figure 0-1. The PAM Web Service Environment

Relationship to Similar or Supplanted Specifications:

The most important relations are to:

- Parlay-X Terminal Status and User Location: Both services deal with information that could be considered part of the user's presence information. Communication abilities can be derived from terminal status information, and the user's placetype can be derived from his location.

- OSA/Parlay PAM: The OSA/Parlay Presence and Availability specification can be considered the big brother of this specification. While ParlayX Presence stays behind OSA PAM in terms of flexibility and power –

especially concerning attributes and management interfaces – it also extends PAM by introducing end-to-end authorization. This specification aims to be mappable to OSA PAM.

- SIP SIMPLE [9]: This specification aims to be mappable to the SIP/SIMPLE architecture.
- XMPP (Jabber): Many principles of this specification have been taken from [10], especially the end-to-end authorization.
- IETF Rich Presence [11]. The set of attributes this document specifies is closely aligned with the IETF's Rich Presence ideas.
- Group Management [13]: Presence of groups is supported by this specification, however their creation and manipulation has to be done using the GM PX web service. In the 3GPP presence context, contact lists and group manipulation is done with the XCAP protocol [15].

5 Namespaces

The PresenceConsumer interface uses the namespace

www.csapi.org/wsdl/parlayx/presence_consumer/v2_0

The PresenceNotification interfaces uses the namespace

www.csapi.org/wsdl/parlayx/presence_notification/v2_0

The PresenceSupplier interfaces uses the namespace

www.csapi.org/wsdl/parlayx/presence_supplier/v2_0

The 'xsd' namespace is used in this document to refer to the XML Schema data types defined in www.w3.org/2001/XMLSchema [5], The use of the name 'xsd' is not semantically significant.

6 Sequence Diagrams

6.1 Interface Flow Overview

The sequence diagram shows the interactions in case both watcher application and presentity are web service clients. Compared to the SIP interactions, the subscription notification is separated from the delivery of presence information itself. Based on the subscription result, the watcher can select the polling or notification mode for presence events. Changes in the authorization of presence attributes are propagated to the watchers via notifySubscription() message, the blocking of a subscription by the presentity are propagated via an endSubscriptionNotification message.

The sequence diagram does not show the internal communication within the presence server. It is assumed that the Presence Consumer and Supplier interfaces are implemented by the same instance. If an implementers of the API find other solutions preferable, he has to take care of the internal communication himself.

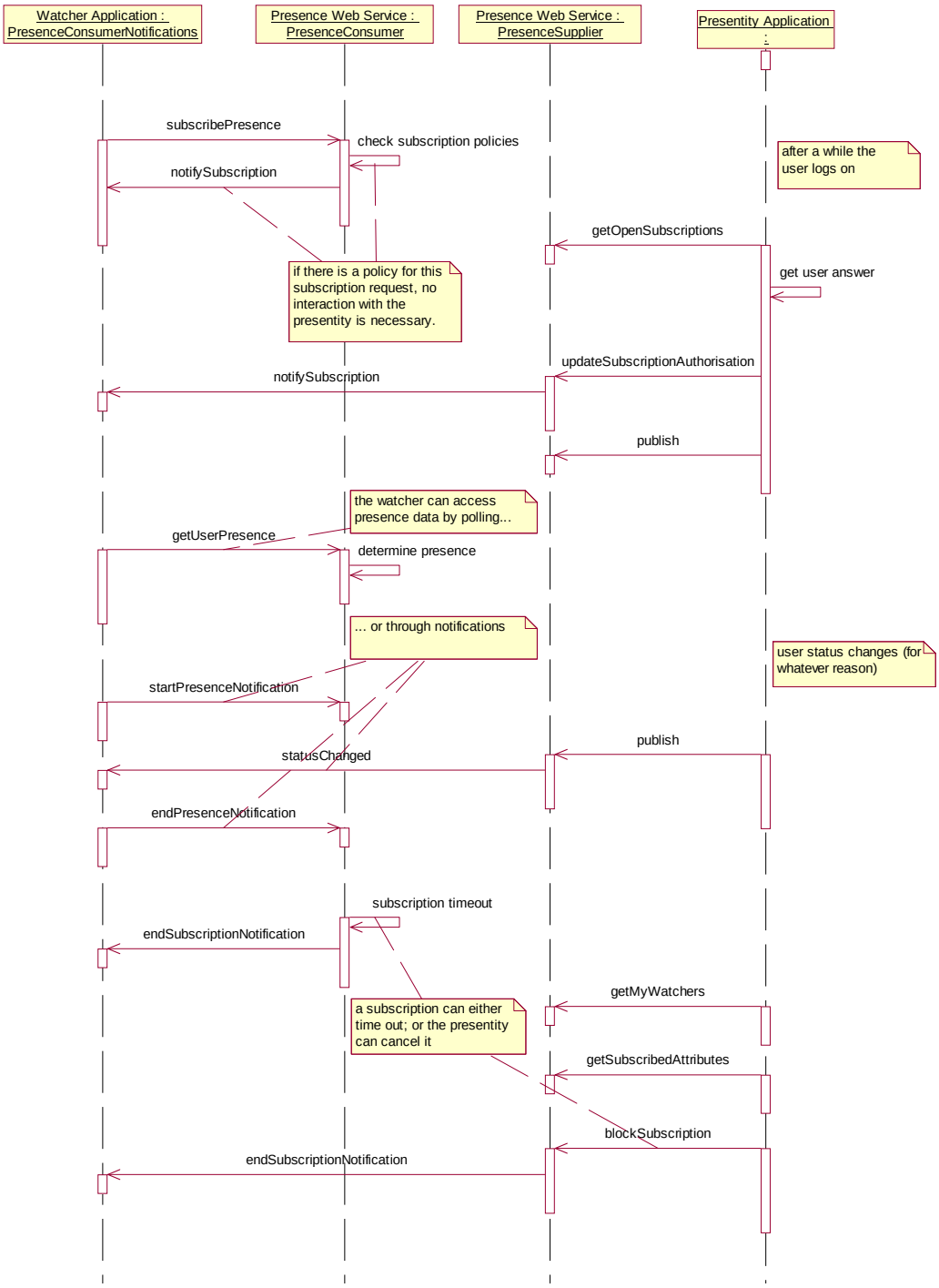


Figure 0-1. Message interaction overview

7 XML Schema Data Type Definition

Presence attributes are inspired by [11].

7.1 PresenceAttributeType Enumeration

The different types of attributes. For each entry in this enumeration there is a separate value type.

Enumeration	Description

Activity	The presentity's activity (available, busy, lunch,…)
Place	At what kind of place the presentity is (home, office,…)
Privacy	The amount of privacy the user wants (public, quiet,..)
Sphere	The user's current environment (work, home)
Communication	The user's means of communication (phone, mail, etc.)
Other	A name – value pair for arbitrary presence information

7.2 ActivityValue Enumeration

This enumeration shows the user's current activity. If the activity is unknown, the attribute value will be `ActivityNone`, meaning the attribute was not set. If the user is doing something not in this list, the value will be set to `ActivityOther`.

Enumeration	Description
ActivityNone	Not set
Available	The user is available for communication.
Busy	The user is busy and is only available for urgent matters.
DoNotDisturb	The user is very busy and does not wish to be disturbed.
OnThePhone	The user is on the phone.
Steering	The user is driving a car / train / airplane etc.
Meeting	The user is in a meeting.
Away	No idea what the user is doing, but he is away.
Meal	The user is eating.
PermanentAbsence	The user is away and will not return for an extended period.
Holiday	The user is on holidays.
Performance	The user is in a theater / concert.
InTransit	The user is in the transit area of an (air)port.
Travel	The user is traveling.
Sleeping	The user is sleeping.
ActivityOther	The user is doing something not in this list.

7.3 PlaceValue Enumeration

This enumeration shows the type of the user's current location. If the place type is unknown, the attribute value will be `PlaceNone`, meaning the attribute was not set. If the user in a place not in this list, the value will be set to `PlaceOther`.

Enumeration	Description
PlaceNone	Not set
Home	The user is at home.

Office	The user is in an office.
PublicTransport	The user is on public transport.
Street	Walking on the street.
Outdoors	Generally outdoors.
PublicPlace	The user is in a public place.
Hotel	The user is in a hotel.
Theater	The user is in a theater or concert.
Restaurant	The user is in a restaurant / bar / etc.
School	The user is at school.
Industrial	The user is in an industrial building.
Quiet	The user is in a quiet area.
Noisy	The user is in a noisy area.
Aircraft	The user is on an aircraft.
Ship	The user is on a ship.
Bus	The user is in a bus.
Station	The user is in a bus- or railway station.
Mall	The user is in a mall.
Airport	The user is in an airport.
Train	The user is in a train.
PlaceOther	The user is in a kind of place not listed here.

7.4 PrivacyValue Enumeration

This enumeration shows the amount of privacy a user currently has. If the privacy is unknown, the attribute value will be `PrivacyNone`, meaning the attribute was not set. If the privacy is not in this list, the value will be set to `PrivacyOther`.

Enumeration	Description
PrivacyNone	Not set
PrivacyPublic	The user is surrounded by other people and cannot discuss openly.
PrivacyPrivate	The user is alone and able to talk openly.
PrivacyQuiet	The user is in a quiet environment and cannot talk at all.
PrivacyOther	None of the other values applies.

7.5 SphereValue Enumeration

This enumeration shows the sphere within which the user acts. If the sphere is unknown, the attribute value will be `SphereNone`, meaning the attribute was not set. If the sphere is not in this list (neither work nor home), the value will be set to `SphereOther`.

Enumeration	Description
SphereNone	Not set
SphereWork	The user is acting within his work sphere, i.e. as a member of his company
SphereHome	The user is acting within his home sphere, i.e. as a private person
SphereOther	The user is acting neither within his work nor within his home sphere.

7.6 CommunicationMeansType Enumeration

This enumeration lists communication means. If the communication attribute referrers to a means not in this list, it will point to `MeansOther`.

Enumeration	Description
Phone	The communication attribute refers to a phone (fixed line or mobile or SIP).
Chat	The communication attribute refers to a chat client.
SMS	The communication attribute refers to an SMS client.
Video	The communication attribute refers to a video phone (fixed line or mobile or SIP).
Web	The communication attribute refers to a web client.
EMail	The communication attribute refers to an e-mail client.
MMS	The communication attribute refers to an MMS client.
MeansOther	The communication attribute refers to any other client.

7.7 CommunicationMeans Structure

This structure describes on way of reaching the presentity.

Element Name	Element Type	Description
Priority	xsd:float	The priority of this communication means. Between 0 and 1, the latter meaning the highest priority.
Contact	xsd:anyURI	The presentity's contact address for this communication means.
Type	CommunicationMeansType	The type of this communication means.

7.8 CommunicationValue Structure

This structure describes the various ways of reaching a presentity.

Element Name	Element Type	Description
Means	CommunicationMeans [0..unbounded]	The different ways of reaching the presentity.

7.9 OtherValue Structure

This structure can be used for storing arbitrary data about a presentity.

Element Name	Element Type	Description
Name	xsd:string	Description of the content.
Value	xsd:string	Attribute content.

7.10 PresenceAttribute Structure

Presence data published by a presentity and retrieved by watchers.

Element Name	Element Type	Description
LastChange	xsd:dateTime	The time and date when the attribute was changed last.
Note	xsd:string	An explanatory note. Optional.
Type	PresenceAttributeType	Determines the type of the value field.
Value	One of the six value types; depends on field "type"	The actual value of the attribute.

This data structure is split into two types in the XSD file: A `PresenceAttribute` contains an `AttributeTypeAndValue`.

7.11 SubscriptionRequest Structure

This structure is returned to the presentity by the PAM web service and contains the requesting watcher and the attributes he wants to subscribe.

Element Name	Element Type	Description
Watcher	xsd:anyURI	The watcher who wants to gain access to data
Attributes	PresenceAttributeType [0..unbounded]	The attributes the watcher wants to see
Application	xsd:string	The name of the application running on behalf of the watcher. Note that this field has solely informative purposes, access rights management is based on watcher id only.

7.12 PresencePermission Structure

The answer from the service to the watcher in the message getSubscriptionStatusResponse.

Element Name	Element Type	Description
Attribute	PresenceAttributeType	The name of the attribute the watcher wanted to subscribe
Decision	xsd:boolean	Whether the presentity accepted the subscription. If no, any further fields should be ignored.

8 Web Service Interface Definition

This API is separated into three interfaces:

PresenceConsumer interface: watcher methods for requesting and subscribing presence data

PresenceNotification interface: is the watcher notification interface for presence events

PresenceSupplier interface: presentity methods for supplying presence data and managing subscriptions

8.1 Interface : PresenceConsumer

Client role: watcher

This set of methods is used by the watcher to obtain presence data. After the subscription to presence data, the watcher can select between a polling mode or a notification mode in order to receive presence data.

8.1.1 Operation : subscribePresence

We assume that the watcher has been previously authenticated, so that his identity is known and can be associated with the subscription at the server.

The presentity is contacted and requested to authorize the watcher. As this process generally involves user interaction there cannot be an immediate response. The watcher is notified with `notifySubscription()`. If the presentity is a group, every member of the group will be contacted for authorization. The watcher will get one notification for each member.

Only after the subscription is completed (and the presentity has allowed access to attributes) may the watcher will get information when he uses `getUserPresence()` or `startPresenceNotification()`.

Note that the SimpleReference contains the correlator string used in subsequent messages to the notification interface.

At this interface level, the subscription has no expiration, although at can be ended from the presentity of the underlying layers (see subscriptionEnded method)

8.1.1.1 Input message : subscribePresenceRequest

Part Name	Part Type	Description
Presentity	xsd:anyURI	A presentity or a group of presentities whose attributes the watcher wants to monitor.
Attributes	PresenceAttributeType [0..unbounded]	The attributes the watcher wants to access. (the same for all the group members). An empty array means subscription of all attributes.
Application	xsd:string	Describes the application the watcher needs the data for.
Reference	common:SimpleReference	The notification interface.

8.1.1.2 Output message : subscribePresenceResponse

Part Name	Part Type	Description
None		

8.1.1.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses – if the presentity address does not exist.

PolicyException from [6].

- POL0006: Groups not allowed

- POL0007: Nested groups not allowed

8.1.2 Operation : getUserPresence

Returns the aggregated presence data of a presentity. Only the attributes which the watcher is entitled to see will be returned. This method does not support group identities.

Before getting these attributes, the watcher has to subscribe to them (see above). The presentity needs not be informed of the access, as he has already consented when the watcher called `requestSubscription()`.

8.1.2.1 Input message : getUserPresenceRequest

Part Name	Part Type	Description
Presentity	xsd:anyURI	The presentity whose data the watcher wants to see.
Attributes	PresenceAttributeType [0..unbounded]	The attributes the watcher wants to see. An empty array means all attributes.

8.1.2.2 Output message : getUserPresenceResponse

Part Name	Part Type	Description
Result	PresenceAttribute [0..unbounded]	The actual presence data.

8.1.2.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses – if the presentity address does not exist.

PolicyException from [6]. The presentity has the possibility to cancel or block a subscription by manipulating the policy rules. The exception informs the watcher about this status change.

- POL0002: Privacy error – if the watcher is not subscribed to the requested data.
- POL0006: Groups not allowed

8.1.3 Operation : startPresenceNotification

The notification pattern with correlation is used in order to be able to correlate the notification events with the request. The attributes represent a subset of the attributes subscribed and can be used as filter.

The watcher sets a notification trigger on certain user presence attribute changes. If the list of attributes is empty, the watcher wants to be notified on all subscribed attributes.

In case the presentity is a group the watcher will receive notifications for every single member of the group. The watcher will only get notifications for those attributes and presentities he subscribed successfully prior to the call. The service will return a list of presentities where the notifications could not be set up.

The presentity needs not be informed of the access, as he has already consented when the watcher called `requestSubscription()`.

Note that the SimpleReference contains the correlator string used in subsequent messages to the notification interface.

8.1.3.1 Input message : startPresenceNotificationRequest

Part Name	Part Type	Description
Presentity	xsd:anyURI	The presentity or group whose attributes the watcher wants to monitor.
Attributes	PresenceAttributeType [0..unbounded]	The attributes the watcher wants to see.
Reference	common:SimpleReference	The notification interface
Frequency	common:TimeMetric	Maximum frequency of notifications (can also be considered minimum time between notifications). In case of a group subscription the service must make sure this frequency is not violated by notifications for various members of the group, especially in combination with `checkImmediate`.
Duration	common:TimeMetric	Length of time notifications occur for, null to use default notification time defined by service policy.
Count	xsd:int	Maximum number of notifications, zero if no maximum
CheckImmediate	xsd:boolean	Whether to check status immediately after establishing notification.

8.1.3.2 Output message : startPresenceNotificationResponse

Part Name	Part Type	Description
Presentities	xsd:anyURI [0..unbounded]	The presentities whose attributes the watcher did not subscribe. Empty if all went fine.

8.1.3.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses – if the presentity URI does not exist.
- SVC0005: Duplicate correlator

PolicyException from [6]. The presentity has the possibility to cancel or block a subscription by manipulating the policy rules. The exception informs the watcher about this status change.

- POL0001: Policy error
- POL0004: Unlimited notifications not supported
- POL0005: Too many notifications requested
- POL0006: Groups not allowed
- POL0007: Nested groups not allowed

8.1.4 Operation : endPresenceNotification

Indicates that the watcher does not want further notifications for a specific notification request (identified by the correlator). Note that the subscription to presence data stays active; the caller of this method remains a watcher and can still use getUserPresence() or reactivate the notifications.

8.1.4.1 Input message : endPresenceNotificationsRequest

Part Name	Part Type	Description
Correlator	xsd:string	The notification the watcher wants to cancel.

8.1.4.2 Output message : endPresenceNotificationResponse

Part Name	Part Type	Description
None		

8.1.4.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.2 Interface : PresenceNotification

This client callback interface is used by the presence consumer interface to send notifications.

8.2.1 Operation : statusChanged

The asynchronous operation is called by the web service when an attribute for which notifications were requested changes.

8.2.1.1 Input message : statusChangedRequest

Part Name	Part Type	Description
Correlator	xsd:string	Identifies the notification request
Presentity	xsd:anyURI	The presentity whose presence status has changed
ChangedAttributes	PresenceAttribute [0..unbounded]	The new presence data

8.2.1.2 Output message : statusChangedResponse

Part Name	Part Type	Description
None		

8.2.1.3 Referenced Faults

None.

8.2.2 Operation : statusEnd

The notifications have ended for this correlator. This message will be delivered when the duration or count for notifications have been completed. This message will not be delivered in the case of an error ending the notifications or deliberate ending of the notifications (using endNotification operation).

8.2.2.1 Input message : statusEndRequest

Part Name	Part Type	Description
Correlator	xsd:string	Correlator provided in request to set up this notification

8.2.2.2 Output message : statusEndResponse

Part Name	Part Type	Description
None		

8.2.2.3 Referenced Faults

None.

8.2.3 Operation : notifySubscription

This asynchronous method notifies the watcher that the server or the presentity handled the pending subscription.

8.2.3.1 Input message : notifySubscriptionRequestt

Part Name	Part Type	Description
Presentity	xsd:anyURI	The presentity whose attributes the watcher wants to monitor
Decisions	PresencePermission [0..unbounded]	Denote the attributes the server/presentity accepted to expose

8.2.3.2 Output message : notifySubscriptionResponse

Part Name	Part Type	Description
none		

8.2.4 Operation : subscriptionEnded

This asynchronous operation is called by the web service to notify the watcher (application) that the subscription has terminated. Typical reasons are a timeout of the underlying SIP soft state subscription (in accordance with [14] and [9]) or the decision of the presentity to block further presence information to that watcher. Since the subscription request has no expiration parameters, the service implementation may provide an inactivity timer that also triggers the subscriptionEnded message.

8.2.4.1 Input message : subscriptionEndedRequest

Part Name	Part Type	Description
Presentity	xsd:anyURI	The presentity to which the subscription has terminated
Reason	xsd:string	Timeout, Blocked

8.2.4.2 Output message : subscriptionEndedResponse

Part Name	Part Type	Description
None		

8.3 Interface : PresenceSupplier

These methods are used by the presentity to supply presence data and manage access to the data by its watchers. We assume that the presentity has been previously authenticated, so that his Identity is known.

8.3.1 Operation : publish

The presentity publishes data about herself. This data will then be filtered by the system and forwarded to the watchers who have ordered notifications.

8.3.1.1 Input message : publishRequest

Part Name	Part Type	Description
Presence	PresenceAttribute [0..unbounded]	The presence attributes the devices of the presentity supports

8.3.1.2 Output message : publishResponse

Part Name	Part Type	Description
None		

8.3.1.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value

PolicyException from [6]

- POL0001: Policy error

8.3.2 Operation : getOpenSubscriptions

Called periodically by the presentity to see if any watchers wants to subscribe to presence data. The client will answer open requests with `updateSubscriptionAuthorization()`.

8.3.2.1 Input message : getOpenSubscriptionsRequest

Part Name	Part Type	Description
None		

8.3.2.2 Output message : getOpenSubscriptionsResponse

Part Name	Part Type	Description
OpenRequests	SubscriptionRequest [0..unbounded]	Any open requests

8.3.2.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error

PolicyException from [6]

- POL0001: Policy error

8.3.3 Operation : updateSubscriptionAuthorization

The presentity answers with this operation to watcher subscriptions for which no authorization policy exists. The answer consists of the attribute and the watcher involved and the permissions for each attribute. Subscription requests that are not answered are assumed pending.

The operation can be used by the presentity to change anytime the authorization for a certain watcher or group to monitor one or several attributes.

If the watcher did not try to subscribe the attribute – i.e. there is not pending subscription from this watcher to an attribute in the decisions array, a PresenceException will be raised and the entire authorization request ignored.

8.3.3.1 Input message : updateSubscriptionAuthorizationRequest

Part Name	Part Type	Description
Watcher	xsd:anyURI	watcher or group of watchers
Decisions	PresencePermission [0..unbounded]	The answers to open requests

8.3.3.2 Output message updateSubscriptionAuthorizationResponse

Part Name	Part Type	Description
None		

8.3.3.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses
- SVC0220: NoSubscriptionRequest

PolicyException from [6]

- POL0001: Policy error

8.3.4 Operation : getMyWatchers

Returns an array of watching identities that are subscribed to the presentity's attributes. They are not necessarily users of the notification system, the mere fact that they are allowed to see the presentity's attributes is enough to be on this list.

8.3.4.1 Input message : getMyWatchersRequest

Part Name	Part Type	Description

None		

8.3.4.2 Output message : getMyWatchersResponse

Part Name	Part Type	Description
Result	xsd:anyURI [0..unbounded]	The list of identities who currently have access to the presentity's attributes.

8.3.4.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error

PolicyException from [6]

- POL0001: Policy error

8.3.5 Operation : getSubscribedAttributes

Returns an array of attributes that a specific watcher has subscribed.

8.3.5.1 Input message : getSubscribedAttributesRequest

Part Name	Part Type	Description
Watcher	xsd:anyURI	The watcher whose subscriptions the presentity wants to know

8.3.5.2 Output message : getSubscribedAttributesResponse

Part Name	Part Type	Description
Result	PresenceAttributeType [0..unbounded]	The attributes the watcher is subscribed to.

8.3.5.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0004: No valid addresses
- SVC0221: Not a watcher – if the URI in the field watcher is not a watcher of the presentity.

PolicyException from [6]

- POL0001: Policy error

8.3.6 Operation : blockSubscription

With this operation the presentity can block entirely the flow of presence information to a certain subscribed watcher by canceling the subscription. The watcher will be notified with an subscriptionEnded() message.

8.3.6.1 Input message : blockSubscriptionRequest

Part Name	Part Type	Description
Watcher	xsd:anyURI	The watcher whose subscriptions the presentity wants to cancel

8.3.6.2 Output message : blockSubscriptionResponse

Part Name	Part Type	Description
None		

8.3.6.3 Referenced Faults

ServiceException from [6]

- SVC0001: Service error
- SVC0002: Invalid input value
- SVC0004: No valid addresses
- SVC0221: Not a watcher – if the URI in the field watcher is not a watcher of the presentity.

PolicyException from [6]

- POL0001: Policy error

9 Fault Definitions

9.1 ServiceException

From [6].

9.1.1 SVC0220 : No subscription request

Message Id	SVC0220
Text	No subscription request from watcher %1 for attribute %2
Variables	%1 – watcher URI %2 – type of attribute, from 0

9.1.2 SVC0221 : Not a watcher

Message Id	SVC0221
Text	%1 is not a watcher
Variables	%1 – watcher URI

10 Service Policies

Name	Type	Description
MaximumNotificationFrequency	common:TimeMetric	Maximum rate of notification delivery (also can be considered minimum time between notifications)
MaximumNotificationDuration	common:TimeMetric	Maximum amount of time a notification may be set up for

DefaultNotificationDuration	common:TimeMetric	Default amount of time a notification will be set up for.
MaximumCount	xsd:int	Maximum number of notifications that may be requested
UnlimitedCountAllowed	xsd:boolean	Allowed to specify unlimited notification count (i.e. specify zero in notification count requested)
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Are nested groups supported in group definitions

Annex A (normative): WSDL of Presence API

The document/literal WSDL representation of this interface specification is compliant to [6] and is contained in text files (contained in archive 29199-14-100-doclit.zip) which accompanies the present document.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040360	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	