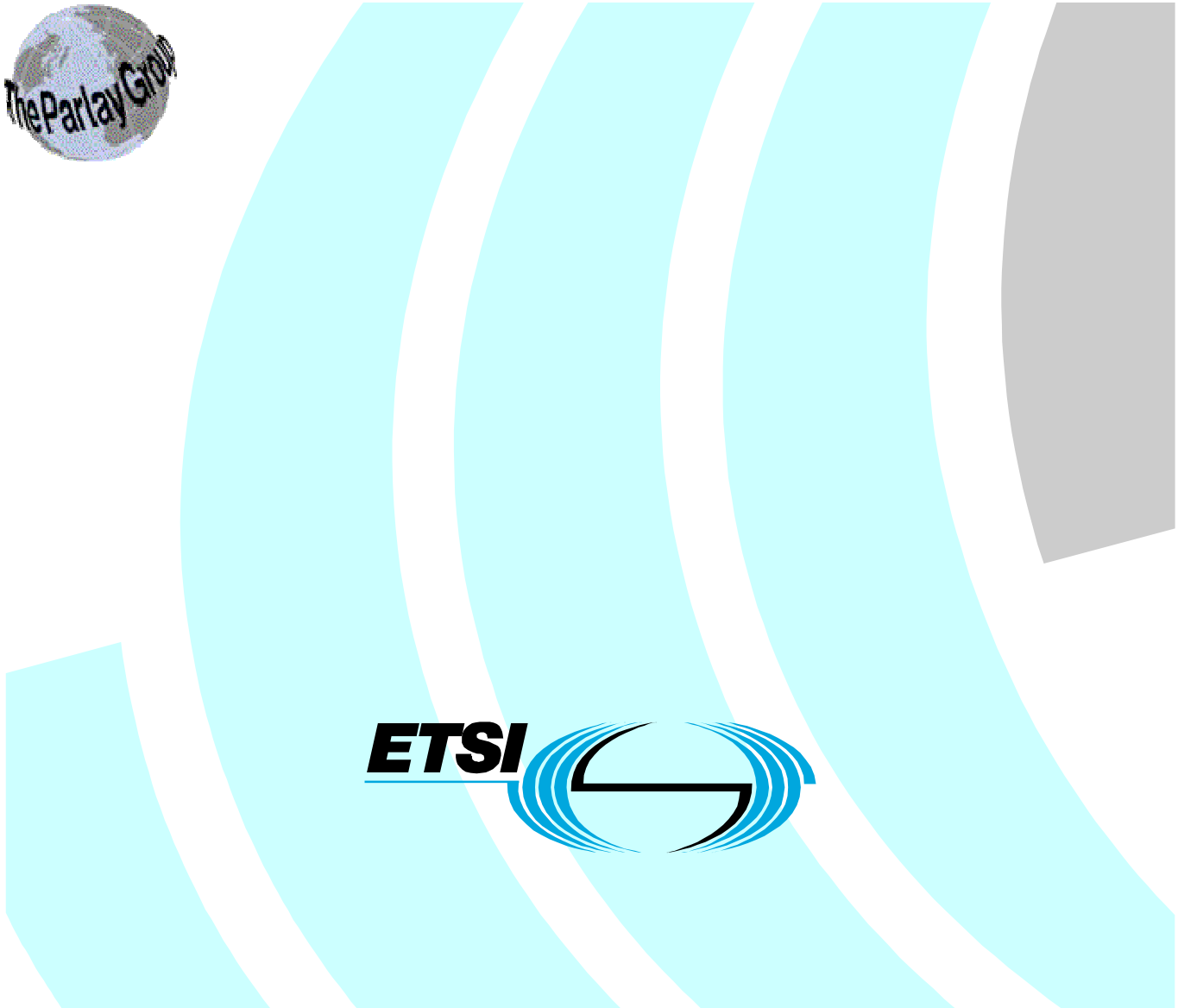
ETSI ES 202 391-1 V1.2.1 (2006-12)

ETSI Standard

Open Service Access (OSA);
Parlay X Web Services;
Part 1: Common
(Parlay X 2)



Reference
RES/TISPAN-01033-01-OSA

Keywords
API, OSA, service

ETSI

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

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

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

Contents

Intellectual Property Rights

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

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

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 1 of a multi-part deliverable covering Open Service Access (OSA); Parlay X Web Services, as identified below:

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".

The present document has been defined jointly between ETSI, The Parlay Group (http://www.parlay.org) and the 3GPP.

The present document forms part of the Parlay X 2.1 set of specifications.

The present document is equivalent to 3GPP TS 29.199-01 V6.3.0 (Release 6).

The Mapping specification of the Parlay X 2 Web Services (ES 202 391) to the Parlay/OSA APIs (ES 203 915 [11]) and, where applicable, to IMS, is also structured as above. A mapping of the Parlay X 2 Web Services is however not applicable for all parts (e.g. part 13), but the numbering of parts is kept.

Table 1: Overview of the Parlay X 2 Web Services and OSA APIs mappings

Parlay X 2 Web Services (ES 202 391 series)		OSA APIs (and IMS) mappings (TR 102 397 series)	
ES 202 391-01	Common	TR 102 397-01	Common mapping
ES 202 391-02	Third party call	TR 102 397-02-1	Generic Call Control mapping
		TR 102 397-02-2	Multi-Party Call Control mapping
ES 202 391-03	Call notification	TR 102 397-03-1	Generic Call Control mapping
		TR 102 397-03-2	Multi-Party Call Control mapping
ES 202 391-04	Short messaging	TR 102 397-04-1	User Interaction mapping
		TR 102 397-04-2	Multi-Media Messaging mapping
ES 202 391-05	Multimedia messaging	TR 102 397-05-1	User Interaction mapping
		TR 102 397-05-2	Multi-Media Messaging mapping
ES 202 391-06	Payment	TR 102 397-06	Charging mapping
ES 202 391-07	Account management	TR 102 397-07	Account Management mapping
ES 202 391-08	Terminal status	TR 102 397-08	Mobility User Status mapping
ES 202 391-09	Terminal location	TR 102 397-09-1	Mobility User Location mapping
		TR 102 397-09-2	Mobility User Location CAMEL mapping
ES 202 391-10	Call handling	TR 102 397-10-1	Generic Call Control & User Interaction mapping
		TR 102 397-10-2	Multi-Party Call Control & User Interaction mapping
		TR 102 397-10-3	Policy Management mapping
ES 202 391-11	Audio call	TR 102 397-11-1	Generic Call Control & User Interaction mapping
		TR 102 397-11-2	Multi-Party Call Control & User Interaction mapping
ES 202 391-12	Multimedia conference	TR 102 397-12	Multi-Media Call Control mapping
ES 202 391-13	*Address list management*	*TR 102 397-13*	*Not Applicable*
ES 202 391-14	Presence	TR 102 397-14-1	Presence & Availability Management mapping
		TR 102 397-14-2	SIP/IMS Networks mapping

1 Scope

The present document is part 1 of the Stage 3 Parlay X 2 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 standardized interface, i.e. the OSA APIs.

The present document specifies the Common aspects of the Parlay X 2 Web Services. The following are defined here:

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

2 References

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

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

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

[1] ETSI TR 121 905: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications (3GPP TR 21.905 version 7.2.0 Release 7)".

[2] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[3] IETF RFC 3966: "The tel URI for Telephone Numbers".

NOTE: Available at: http://www.ietf.org/rfc/rfc3966.txt.

[4] IETF RFC 3261: "SIP: Session Initiation Protocol".

NOTE: Available at: http://www.ietf.org/rfc/rfc3261.txt.

[5] WS-I Basic Profile Version 1.0: "Final Material".

NOTE: Available at: http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html.

[6] W3C Note (15 March 2001): "Web Services Description Language (WSDL) 1.1".

NOTE: Available at: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[7] OASIS Standard 200401 (March 2004): "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)".

NOTE: Available at: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[8] W3C Recommendation (12 February 2002): "XML-Signature Syntax and Processing".

NOTE: Available at: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.

[9] ISO 4217: "Codes for the representation of currencies and funds".

[10] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

NOTE: Available at: http://www.ietf.org/rfc/rfc3986.txt.

[11] ETSI ES 203 915 (series): "Open Service Access (OSA); Application Programming Interface (API)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

application: computer program that accesses a Web Service

SOAP: not an acronym, protocol used for XML messaging

Web Service: software system designed to support interoperable machine-to-machine interaction over a network

Web Service Provider: entity which provides Web Services interfaces to capabilities offered

Web Service Requester: entity which operates Applications that access Web Services

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 121 905 [1] and the following apply:

3GPP	Third Generation Partnership Project
ETSI	European Telecommunications Standards Institute
IT	Information Technology
OASIS	Organization for the Advancement of Structured Information Standards
OSA	Open Service Access
RFC	Request For Comment
SIP	Session Initiation Protocol
UDDI	Universal Description Discovery and Integration
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 [5], 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 [5].

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 [5].

4.2 Web Service interface definitions

All Parlay X 2 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 [6] and be conformant to the WSDL use defined in WS-I Basic Profile [5].

See clause 12 for detailed information on the WSDL style to be followed by Parlay X 2 Web Services.

4.3 Security for Parlay X 2 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 [7].

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 [5].

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 [8].

4.4 XML data types

Where possible standard XML Schema data types are used, as defined in clause 3 (Built-in datatypes) in XML Schema [2].

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 3966 [3] for 'tel:' addresses, RFC 3261 [4] for 'sip:' addresses or the definition given below for shortcodes or aliased addresses. Optional additions to the address portion of these URI definitions MUST NOT be considered part of the address accepted by the Parlay X 2 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 3966 [3], Parlay X 2 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 3966 [3] or RFC 3261 [4]).

Shortcodes are short telephone numbers, usually 4 to 6 digits in length reserved for telecom service providers' own functionality. They shall be differentiated from national addresses by the use of a 'short:' rather than 'tel:' URI scheme. The short code defined in the URI consists of a string of digits with no non-digit characters.

Support for aliases in addresses is provided by use of the URI defined in RFC 3986 [10]. This allows for arbitrary data to be submitted to the Parlay X Web Service interface. The following is an example of how this could be applied:

<uri scheme>:<generic syntax>

An alias is generally a relatively short character string that holds a scrambled address such that only the application identified in the URI can expand it.

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 [9].
- 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/parlayx/common/v2_1

The namespace for the common faults is:

- http://www.csapi.org/wsdl/parlayx/common/v2_0/faults

The "xsd" namespace is used in the present document to refer to the XML Schema data types defined in XML Schema [2]. 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 value	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	Optional	Description
metric	TimeMetrics	No	Metric to use for time measurement
units	xsd:int	No	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	Optional	Description
description	xsd:string	No	Description text to be use for information and billing text
currency	xsd:string	Yes	Currency identifier as defined in ISO 4217 [9]
amount	xsd:decimal	Yes	Amount to be charged
code	xsd:string	Yes	Charging code, referencing a contract under which the charge is applied

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	Optional	Description
messageId	xsd:string	No	Message identifier (take from fault definitions)
text	xsd:string	No	Message text, with replacement variables marked with %#
variables	xsd:string [0..unbounded]	Yes	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	Optional	Description
endpoint	xsd:anyURI	No	Endpoint address
interfaceName	xsd:string	No	Name of interface
correlator	xsd:string	No	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 2 Web Services specification use. Numbers from "1000" to "9999" may be used by third parties.

Element name	Element type	Optional	Description
messageId	xsd:string	No	Message identifier, with prefix SVC
text	xsd:string	No	Message text, with replacement variables marked with %#
variables	xsd:string [0..unbounded]	Yes	Variables to substitute into Text string

10.1.1 SVC0001: Service error

Element name	Description
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

Element name	Description
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

Element name	Description
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

Element name	Description
messageId	SVC0004
text	No valid addresses provided in message part %1
variables	%1 - message part

10.1.5 SVC0005: Duplicate correlator

Element name	Description
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

Element name	Description
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

Element name	Description
messageId	SVC0007
text	Invalid charging information
variables	None

10.1.8 SVC0008: Overlapping Criteria

Name	Description
message Id	SVC0008
text	Overlapped Criteria %1
variables	%1 Message part with the overlapped criteria

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 2 Web Services specification use. Numbers from "1000" to "9999" may be used by third parties.

Element name	Element type	Optional	Description
messageId	xsd:string	No	Message identifier, with prefix POL
text	xsd:string	No	Message text, with replacement variables marked with %#
variables	xsd:string [0..unbounded]	Yes	Variables to substitute into Text string

10.2.1 POL0001: Policy error

Element name	Description
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

Element name	Description
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

Element name	Description
messageId	POL0003
text	Too many addresses specified in message part %1
variables	%1 - message part

10.2.4 POL0004: Unlimited notifications not supported

Element name	Description
messageId	POL0004
text	Unlimited notification request not supported
variables	None

10.2.5 POL0005: Too many notifications requested

Element name	Description
messageId	POL0005
text	Too many notifications requested
variables	None

10.2.6 POL0006: Groups not allowed

Element name	Description
messageId	POL0006
text	Group specified in message part %1 not allowed
variables	%1 - message part

10.2.7 POL0007: Nested groups not allowed

Element name	Description
messageId	POL0007
text	Nested group specified in message part %1 not allowed
variables	%1 - message part

10.2.8 POL0008: Charging not supported

Element name	Description
messageId	POL0008
text	Charging is not supported
variables	None

10.2.9 POL0009: Invalid frequency requested

Element name	Description
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 2 Web Services.

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

11 Service policies

Not applicable.

12 WSDL usage and style

Parlay X 2 Web Services definitions:

- SHALL specify services using document forms as described in clause 12.1.
- SHALL use namespaces as defined in clause 12.2.
- SHALL follow the authoring style as defined in clause 12.3.
- SHALL follow data type definitions as defined in clause 12.4.
- SHALL define messages and interfaces as defined in clause 12.5 using document/literal definitions.
- SHALL define bindings and services as defined in clause 12.6 using document/literal definitions.

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 [6]). 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".

The present 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 the present 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.

The present 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 [6]) definitions. One interface definition is included in each document. The present document may import *Type Definition Documents* and *Shared Faults Documents*. The present document may be used for a variety of *Service Bindings Documents* without change.

The document name for the present 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 the present 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. The present document imports one *Service Interface Document*.

The document name for the present 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 the present 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 clause 12.2.3 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 2 Web Services

For Parlay X 2 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 Namespace usage

12.2.4.1 Why namespace versions are used

Maintaining a version number as part of the namespace enables multiple versions of a specification to be identified easily, both by human inspection (reading namespace information) and by machine inspection (parsing namespace).

In addition to the version information being contained in the namespace, the Parlay X Web Services WSDL documents also incorporate this same version number in the document file name.

These two mechanisms enable each specification document version to be uniquely identified, ensuring correct composition of documents even when multiple versions are present in a system.

12.2.4.2 When namespace versions are changed

When a specification document, or one of its dependent documents, changes then the version for the specification document is incremented. Incrementing of the major version or minor version number is dependent on the nature of the change (typically major version number changes at a release cycle, minor version number changes within a release cycle).

For example, if a specification has a types definition document and an interface definition document, then:

- If the types definitions document is updated, its version will be incremented. Since the interface definition is dependent on the types definition document, its version will be incremented as well.

- If the interface definition document is updated, but there are no changes to the types definition document, then only the interface definition document version is incremented (since the types definition document is not dependent on the interface definition document).

For common documents that are used across multiple specifications, a change in the common document will require updating the specification documents that are dependent on the common document (to update the reference to the common document) and thus their document versions will be incremented as well.

12.2.4.3 Benefit of managing namespace versions

Two primary benefits are realized by managing namespace versions.

- Clearly identified specification documents for the specification reader, developer or machine (for discovery).
- Possibility to have coexisting implementations of multiple versions of the same specification, since all artefacts are isolated and uniquely identifiable.

12.2.5 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.

12.2.6 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.

12.2.7 WSDL and Schema namespaces

Namespaces are defined for the WSDL and Schema elements that are defined within the present document, and for those that are referenced by elements in the present 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 the present 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
```

12.2.8 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.

12.2.9 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 clause 12.3.3. 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 the present 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" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Work" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Mobile" type="example_xsd:MobileNumber" minOccurs="0" maxOccurs="1"/>
   </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. A correlator should be unique for the service receiving the reference. A means shall be used to guarantee that correlators are unique, incorporating for example message based identifiers or a means agreed between the application and service instance. Note that when requests are retried after certain failure responses that solicit an amendment to a request (for example, a challenge for authentication), these retried requests are not considered new requests, and therefore do not need a new correlator. Implementations may use the form "localid@host". Correlators are case-sensitive and are simply compared byte-by-byte. No provisioning or human interface is required for the selection of the correlator.

```
<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.

The present 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 [5].

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 the present document and is contained in text files (contained in archive es_20239101v010201p0.zip) which accompany the present document.

History

Document history		
V1.1.1	March 2005	Publication
V1.2.1	October 2006	Membership Approval Procedure MV 20061215: 2006-10-17 to 2006-12-15
V1.2.1	December 2006	Publication