RESTful bindings for Parlay X Web Services - Common

Approved Version 1.1 – 24 Jul 2012

Open Mobile Alliance

OMA-TS-ParlayREST_Common-V1_1-20120724-A

Contents

1. SCOPE 5
2. REFERENCES 6
2.1 NORMATIVE REFERENCES 6
2.2 INFORMATIVE REFERENCES 6
3. TERMINOLOGY AND CONVENTIONS 7
3.1 CONVENTIONS 7
3.2 DEFINITIONS 7
3.3 ABBREVIATIONS 7
4. INTRODUCTION 8
4.1 VERSION 1.0 8
4.2 VERSION 1.1 8
5. COMMON CONSIDERATIONS FOR PARLAYREST 9
5.1 NAMESPACES 9
5.2 ERROR RECOVERY DURING RESOURCE CREATION 9
5.3 ENCODING AND SERIALIZATION DETAILS FOR MIME FORMAT 9
6. SHARED DATA TYPE DEFINITIONS 11
6.1 CHARGING 11
6.1.1 Charging data type 11
6.2 COMMON DATA TYPES 12
6.2.1 Enumeration: NotificationFormat 12
6.2.2 Void 12
6.2.3 Void 12
6.2.4 Type: ChargingInformation 12
6.2.5 Type: CallbackReference 12
6.2.6 Type: ResourceReference 13
6.2.7 Type: Link 13
6.2.8 Type: RequestError 13
6.2.9 Type: ServiceException 13
6.2.10 Type: PolicyException 14
6.2.11 Type: ServiceError 14
6.2.12 Enumeration: RetrievalStatus 14
6.2.13 Type: CallSessionInformation 14
6.2.14 Type: CallParticipantInformation 16
6.2.15 Type: MediaInfo 18
6.2.16 Type: LanguageString 18
6.2.17 Enumeration: Media 18
6.2.18 Enumeration: MediaDirection 18
6.2.19 Enumeration: CallParticipantStatus 18
6.2.20 Enumeration: CallParticipantTerminationCause 19
6.3 SERVICE AND POLICY EXCEPTIONS 19
6.3.1 Service exception 19
6.3.2 Policy exception 20
6.3.3 HTTP Response Codes in response to Notifications 20
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 21
A.1 APPROVED VERSION HISTORY 21
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS 22
APPENDIX C. PRINCIPLES FOR DEFINING REST BINDINGS FOR PARLAY X WEB SERVICE APIS (INFORMATIVE) 23

Tables

Table 1: NotificationFormat Values....12
Table 2: ChargingInformation Structure....12
Table 3: CallbackReference Structure....13
Table 4: ResourceReference Structure....13
Table 5: Link Structure....13
Table 6: RequestError....13
Table 7 ServiceException....14
Table 8: PolicyExceptionResponse Codes....14
Table 9: ServiceError....14
Table 10: RetrievalStatus....14
Table 11: CallSessionInformation....16
Table 12: CallParticipantInformation....17
Table 13: MediaInfo....18
Table 14: Media....18
Table 15: MediaDirection....18
Table 16: CallParticipantStatus....19
Table 17: CallParticipantTerminationCause....19

1. Scope

The scope of this specification is to specify an HTTP protocol binding for the set of Parlay X Web Services specifications in OMA, using REST architectural style.The specification defines an HTTP protocol binding for an abstract API, based on existing OMA enablers. The material in this TS is specific to the ParlayREST enabler.

2. References

2.1 Normative References

[Common_TS]	"Common definitions and specifications for OMA REST interfaces", Open Mobile Alliance™, OMA-TS-REST_Common-V1_0, URL: http://www.openmobilealliance.org/
[HTML FORMS]	"HTML Forms", W3C Recommendation, URL:http://www.w3.org/TR/html401/interact/forms.html
[ISO4217]	"ISO 4217 currency names and code elements", URL: http://www.iso.org/
[JSON]	"The application/json Media Type for JavaScript Object Notation (JSON)", D. Crockford, July 2006, URL: http://www.ietf.org/rfc/rfc4627.txt
[ParlayX_Common]	"Open Service Access (OSA); Parlay X web services; Part 1: Common", Release 8, Third Generation Partnership Project, URL: http://www.3gpp.org/ftp/Specs/html-info/29-series.htm
[PSA]	"Reference Release Package for Parlay Service Access", Open Mobile Alliance™, OMA-ERP-PSA-V1_0, URL: http://www.openmobilealliance.org/
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL: http://www.ietf.org/rfc/rfc2119.txt
[RFC2388]	"Returning Values from Forms: multipart/form-data", L. Masinter, August, 1998, URL:http://www.ietf.org/rfc/rfc2388.txt
[RFC2616]	"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et. al, June 1999, URL: http://www.ietf.org/rfc/rfc2616.txt
[RFC3261]	"SIP: Session Initiation Protocol", J. Rosenberg, et. Al, June 2002, URL: http://www.ietf.org/rfc/rfc3261.txt
[RFC3986]	"Uniform Resource Identifier (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, January 2005, URL: http://www.ietf.org/rfc/rfc3986.txt
[RFC3966]	"The tel URI for Telephone Numbers", H. Schulzrinne, December 2004, URL: http://www.ietf.org/rfc/rfc3966.txt
[RFC4122]	"A Universally Unique IDentifier (UUID) URN Namespace", P. Leach, M. Mealling, R. Salz, July 2005, URL: http://www.ietf.org/rfc/rfc4122.txt
[SCRRULES]	"SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL: http://www.openmobilealliance.org/
[W3C-XML11]	W3C XML 1.1 Specification, URL: http://www.w3.org/TR/xml11/
[XMLSchema1]	W3C Recommendation, XML Schema Part 1: Structures Second Edition, URL: http://www.w3.org/TR/xmlschema-1/
[XMLSchema2]	W3C Recommendation, XML Schema Part 2: Datatypes Second Edition, URL: http://www.w3.org/TR/xmlschema-2/

2.2 Informative References

[OMADICT]	"Dictionary for OMA Specifications", Version 2.8, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_8, URL: http://www.openmobilealliance.org/
[REST_WP]	"White Paper on Guidelines for REST API specifications", Open Mobile Alliance™, OMA-WP-Guidelines-for-REST-API-specifications, URL:http://www.openmobilealliance.org/
[XML2JSON]	Open source XML to JSON conversion tool URL: http://forge.morfeo-project.org

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purpose of this TS, all definitions from the OMA Dictionary apply [OMA-DICT].

[N/A]	[N/A]

3.3 Abbreviations

API	Application Programming Interface
DNS	Domain Name Server
HTTP	Hypertext Transfer Protocol
ID	Identifier
IP	Internet Protocol
JSON	JavaScript Object Notation
OMA	Open Mobile Alliance
PLMN	Public Land Mobile Network
PSA	Parlay Service Access
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
XML	Extensible Markup Language
XSD	XML Schema Definition

4. Introduction

To ensure consistency for developers using the ParlayREST enabler, this “Common” technical specification aims to contain all items that are common across all HTTP protocol bindings using REST architectural style for the various individual interface definitions, such as namespaces, naming conventions and fault definitions. In addition, data types that are shared between two or more protocol bindings are included in this specification as well.

4.1 Version 1.0

This version of the ParlayREST Common Technical Specification contains common namespaces, naming conventions and fault definitions, as well as shared data types for ParlayREST V1_0.

The ParlayREST TSs for version 1.0 addresses the following enablers:

- Short Messaging, as defined in PSA V1_0 [PSA]
- Multimedia Messaging, as defined in PSA V1_0 [PSA]
- Terminal Location, as defined in PSA V1_0 [PSA]
- Payment, as defined in PSA V1_0 [PSA]

4.2 Version 1.1

On top of the content described in the section above, this version of the ParlayREST Common Technical Specification contains further shared data types for the following ParlayREST enablers:

- Third Party Call, as defined in PSA V1_0 [PSA]
- Call Notification, as defined in PSA V1_0 [PSA]
- Audio Call, as defined in PSA V1_0 [PSA]
- Terminal Status, as defined in PSA V1_0[PSA]

5. Common Considerations for ParlayREST

All the common definitions and specifications that apply to any generic OMA REST interface [Common_TS] apply to the ParlayREST interfaces as well. In addition, the following definitions and specifications apply.

5.1 Namespaces

The namespace for the common data types is:

urn:oma:xml:rest:common:1

The 'xsd' namespace is used in the present document to refer to the XML Schema data types defined in XML Schema [XMLSchema1, XMLSchema2]. The use of the name 'xsd' is not semantically significant.

5.2 Error recovery during resource creation

The following mechanism allows recovery from communication failures that can occur during resource creation using POST.

The client MAY (and in some cases SHOULD) include in the parameter set of the resource creation request the "clientCorrelator" field which uniquely maps to the resource to be created.

Note that this allows the client to retry a resource-creating request for which it did not receive an answer due to communication failure, and prevents the duplicate creation of resources on the server side in case of such retry. Note further that depending on the deployment (e.g. Network Address Translation, Proxies), the server might or might not be able to distinguish between different clients.

It is therefore RECOMMENDED that the client generates the value of the “clientCorrelator” in such a way that collisions (i.e. two unrelated requests use the same “clientCorrelator” value) are impossible or at least highly improbable. The way this is achieved is out of scope of this specification, however, it is pointed out that for example UUID [RFC4122] provides a way to implement such a scheme.

In case the server receives a “clientCorrelator” value in a resource-creating POST request, it SHALL do the following:

- in case the request contains a “clientCorrelator” value that has not been used yet to create a resource, the server SHALL create the resource and respond with "201 Created", as above.
- in case the request contains a “clientCorrelator” value that has already been used to create a resource, the server responds as follows:
- in case this is a valid repeated attempt by the same client to create the same resource, the server SHALL respond with "200 OK", and SHALL return a representation of the resource.
- otherwise, it SHALL respond with "409 Conflict", in this case indicating a clientCorrelator conflict. In such case, the client can retry the request using a new “clientCorrelator” value.

5.3 Encoding and Serialization Details for MIME format

A MIME multipart message (used for instance in the MMS API) consists of several parts:

- The root structure (e.g. InboundMMSMesssage), which is a ParlayREST data structure, expressed in the different possible formats (xml, json or even in url-encoded format for requests). This part conveys the ParlayREST resource parameters.
- The multimedia contents or attachments as MIME body parts, within the HTTP request or response. They include all contents, both plain text as well as other content types (images, videos, etc).

In the ParlayREST APIs, for simplicity purposes and better suitability to the internet developer community and browsers, multipart/form-data [RFC2388] and [HTML FORMS] will be used instead of multipart/related. This implies that:

1. **Root fields** as described above SHALL be included as a single form field with a MIME body with:

    Content-Disposition: form-data; name=”root-fields”

    Content-Type: <Corresponding Content type>

   Allowed content types for the root fields are:

   - application/xml
   - application/json
   - application/x-www-form-urlencoded

2. **Multimedia contents** (text, images, etc.) SHALL be included using one of the following two options:

   a. When the message contains *only one content item:* By including a MIME body with:

      Content-Disposition: form-data; name="attachments", filename="<Name of the message content>"

      Content-Type: <Corresponding Content-Type>

   b. When the message contains *more than one content item*: By including a form-field with a MIME body with:

      Content-Disposition: form-data; name="attachments"

      Content-Type: multipart/mixed

   Then, every one of the possible message contents SHALL be included as subparts, with:

      Content-Disposition: attachment; filename="<Name of the message content>" Content-Type: <Corresponding Content-Type>

3. For every MIME body part and subparts, it is possible to include other parameters (Content-Description, Content-Transfer-Encoding, Content-ID), etc.

6. Shared Data Type Definitions

This section contains data type definitions which are shared among two or more REST protocol bindings.

6.1 Charging

This section deals with in-band charging, i.e. passing charging data as part of the API request. To enable this capability to be provided across a variety of services in a consistent manner, the information to be provided in the message for charging information is defined as a common charging data type.

6.1.1 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" minOccurs="1" maxOccurs="unbounded"/>
    <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 Service provides this information:

- *"description"* is an array of text. The first entry of a list 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.

- When more than one entry is provided, the rest should be references to individual operations relevant to the charging. Reference should be set to a value provided in a response message to the operation as a unique identifier to correlate individual operation.

- *"currency"* in which the charge is to be applied. Values for the currency field are defined by [ISO4217].

- *"amount"* 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 Service Provider.

The charging information provided may not be acceptable to the Service Provider. For example, the Service Provider may limit the amount that may be specified for a particular Service or for a particular Service Requester. If the information provided is not acceptable, an appropriate fault message may be returned to the requester (SVC0007 and POL0012 are defined as a generic charging fault, The 'SVC' and 'POL' service exceptions are defined in [ParlayX_Common]).

Especially in case of charging operation such as creating a charge or refund, it is strongly recommended to convey a list of relevant operations related to charging over a description part as described above.

This is useful especially when a charging operation is performed after a certain set of operations.

Some of the services may be meaningful to the user only when a certain set of operations is completed. In that case, service provider may want to charge a user only upon a completion of the entire process, instead of charging per operation. Also, service provider may want to control the actual amount of charging depending on a certain condition, e.g., service usage volume, independent of the volume control provided by the network operators. This is also the case where it is preferable to perform charging operation after a completion of certain set of operations. In these cases where a service provider charges a user for the consumption of a certain service, the service provider is recommended to provide the references to the individual operations performed as evidences. This information can be referenced by the relevant entities to ensure the validity of charging when necessary.

It should be noted that this is for a service provider to provide a list of evidences of their direct use of operations. Any mapping of underlying operations performed internally in the operator must be performed by the operator if necessary. How to maintain the consistency between the information kept at service provider and the operators is out of scope. Also, charging aspects which do not relate to any operations are not covered.

6.2 Common data types

6.2.1 Enumeration: NotificationFormat

List of notification format values.

Enumeration	Description
XML	Notification about new inbound message would use XML format in the POST request
JSON	Notification about new inbound message would use JSON format in the POST request

Table 1: NotificationFormat Values

6.2.2 Void

6.2.3 Void

6.2.4 Type: ChargingInformation

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

Element	Type	Optional	Description
description	xsd:string [1..unbounded]	No	An array of description text to be use for information and billing text
currency	xsd:string	Yes	Currency identifier as defined in [ISO4217]
amount	xsd:decimal	Yes	Amount to be charged/refunded/reserved. The amount to be charged/refunded/reserved appears either directly in the amount-field or as code in the code-field. If both these two fields are missing or empty a service exception (SVC0007) will be thrown.
code	xsd:string	Yes	Charging code, referencing a contract under which the charge is applied

Table 2: ChargingInformation Structure

6.2.5 Type: CallbackReference

An application can use the CallbackReference data structure to subscribe to notifications.

If a parameter *callbackData* has been passed in a particular subscription, the server MUST copy it into each notification which is related to that particular subscription.

Element	Type	Optional	Description
notifyURL	xsd:anyURI	No	Notify Callback URL
callbackData	xsd:string	Yes	Data the application can register with the server when subscribing to notifications, and that are passed back

			unchanged in each of the related notifications. These data can be used by the application in the processing of the notification, e.g. for correlation purposes.
notificationFormat	NotificationFormat	Yes	Default: XML Application can specify format of the resource representation in notifications that are related to this subscription. The choice is between {XML, JSON}

Table 3: CallbackReference Structure

Note: In case the application requires correlating notifications to the related subscription, it can either submit a different *notifyURL* in each subscription, or use the optional *callbackData* parameter as a correlator.

6.2.6 Type: ResourceReference

Element	Type	Optional	Description
resourceURL	xsd:anyURI	No	The URL that addresses the resource. SHALL NOT be included in POST requests, MUST be included in responses to any HTTP method that returns an entity body, and in PUT requests.

Table 4: ResourceReference Structure

The *resourceReference* element of type *ResourceReference* is defined as a root element in the XSD.

6.2.7 Type: Link

Attribute	Type	Optional	Description
rel	xsd:string	No	Describes the relationship between the URI and the resource
href	xsd:anyURI	No	URI

Table 5: Link Structure

An element of type *Link* can be provided by the server to point to other resources that are in relationship with the resource. The *rel* attribute is a string. The possible values for the string are defined in each REST enabler. *Rel* and *href* are realized as attributes in the XSD.

6.2.8 Type: RequestError

Element	Type	Optional	Description
link	Link[0..unbounded]	Yes	Link to elements external to the resource
serviceException	ServiceException	Choice	Exception Details
policyException	PolicyException	Choice	Exception Details

Table 6: RequestError

A requestError element of type *RequestError* is defined as a root element in the XSD.

6.2.9 Type: ServiceException

Element	Type	Optional	Description
messageId	xsd:string	No	Message identifier, with prefix SVC
text	xsd:string	No	Message text, with replacement variables marked with %*n*, where *n* is an index into the list of <variables> elements, starting at 1

variables	xsd:string [0..unbounded]	Yes	Variables to substitute into Text string

Table 7 ServiceException

6.2.10 Type: PolicyException

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 %*n*, where *n* is an index into the list of <variables> elements, starting at 1
variables	xsd:string [0..unbounded]	Yes	Variables to substitute into Text string

Table 8: PolicyExceptionResponse Codes

6.2.11 Type: ServiceError

In a response to a request, ServiceError is used when an operation involving multiple items fails for only some of the items, whereas ServiceException is used where the entire operation fails.

In notifications, ServiceError is always used to indicate a notification termination or cancellation.

Element	Type	Optional	Description
messageId	xsd:string	No	Message identifier, either with prefix SVC or with prefix POL
text	xsd:string	No	Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1
variables	xsd:string [0..unbounded]	Yes	Variables to substitute into text string

Table 9: ServiceError

6.2.12 Enumeration: RetrievalStatus

Enumeration	Description
Retrieved	Data retrieved. Current data is provided
NotRetrieved	Data not retrieved, current data is not provided (does not indicate an error, no attempt may have been made). Note that this field is useful in case a list of addresses are requested, some items could be marked as "NotRetrieved" in case retrieval could not be attempted for some reason, e.g. to avoid time outs
Error	Error retrieving data

Table 10: RetrievalStatus

6.2.13 Type: CallSessionInformation

This type defines information about a call session.

Element	Type	Optional	Description

participant	CallParticipantInformation [1..unbounded]	No	The participants in this call. The first element in this list is considered to denote the "A-Party" (i.e. originator) of the call session.
participantAnnouncement	xsd:string	Yes	A reference to an announcement to be played to the participants listed in the "participant" element upon joining the call. If the "originatorAnnouncement" element is set in addition, that announcement SHALL be played to the first participant instead of this announcement. If this element is not set by the client, no announcement SHALL be played. The server SHALL understand the value "default" and play a default announcement if this value is provided as the content of this element. Further values are allowed as long as they are meaningful to the server, but these values are out of scope of this specification.
originatorAnnouncement	xsd:string	Yes	A reference to an announcement that is played to the first participant listed in the "participant" element. The first participant is viewed as the originator. This element SHALL NOT be provided if the element "participantAnnouncement" The content of the field is out of scope for this specification but is assumed to a value that is meaningful to the server.
callbackReference	common:CallbackReference	Yes	Endpoint on which to notify the client of events related to this call session, and related parameters. The client is NOT REQUIRED to set this element, but instead MAY choose to ignore, to poll for, or to use the notification framework defined in [REST_TS_CallNotif] to obtain information about the call and its participants.
terminated	xsd:boolean	Yes	Indicates whether the session has

			been terminated (true) or is active (false). Default value is false.
charging	common:ChargingInformation	Yes	Charge to apply to the call session
mediaInfo	MediaInfo [0..unbounded]	Yes	It identifies one or more media type(s) for the call, i.e. the media type(s) to be applied to the participants in the call session. If the parameter is omitted, the media type(s) are negotiated by the underlying network.
changeMediaNotAllowed	xsd:boolean	Yes	If true, no call participant (user) in the call will be permitted to change media type during the call. If false the end user may change media type after the call is established as no network protection mechanism is set up to prevent participant (end user) initiated change of media type. Default: true
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship with the resource
clientCorrelator	xsd:string	Yes	A correlator that the client MAY use to tag this particular resource representation during a request to create a resource on the server. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
resourceURL	xsd:anyURI	Yes	Self-reference. SHALL NOT be included in POST requests, MUST be included in responses to any HTTP method that returns an entity body, and in PUT requests.

Table 11: CallSessionInformation

Note that the clientCorrelator is used for purposes of error recovery as specified in section 5.2, and internal client purposes. The server is NOT REQUIRED to use the clientCorrelator value in any form in the creation of the URL of the resource. Section 5.2 of provides a recommendation regarding the generation of the value of this field.

Note: This type can be shared by multiple specifications, which can each define their own instances of root or child elements of this type.

6.2.14 Type: CallParticipantInformation

This type defines information about a call participant.

Element	Type	Optional	Description
participantAddress	xsd:anyURI	No	Address of the user. Corresponds to CallParticipantIdentifier in Parlay X.
participantName	xsd:string	Yes	The name of the participant.
participantStatus	CallParticipantStatus	Yes	Indicates the current status of the participant in the call.
startTime	xsd:dateTime	Yes	Indicates the time when the call participant was added to the call. This field SHALL be present when CallParticipantStatus <> CallParticipantInitial).
duration	xsd:int	Yes	Indicates the duration of the call participant's involvement in the call, expressed in seconds. This field SHALL be present when CallParticipantStatus = CallParticipantTerminated.
terminationCause	CallParticipantTerminationCause	Yes	It indicates the cause of the call participants termination from the call. This field SHALL be present when CallParticipantStatus = CallParticipantTerminated.
mediaInfo	MediaInfo [0..unbounded]	Yes	When applicable, it indicates the media currently used in the call for the identified participant.
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship with the resource
clientCorrelator	xsd:string	Yes	A correlator that the client MAY use to tag this particular resource representation during a request to create a resource on the server. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
resourceURL	xsd:anyURI	Yes	Self-reference. SHALL NOT be included in POST requests, MUST be included in responses to any HTTP method that returns an entity body, and in PUT requests.

Table 12: CallParticipantInformation

Note that the clientCorrelator is used for purposes of error recovery as specified in section 5.2, and internal client purposes. The server is NOT REQUIRED to use the clientCorrelator value in any form in the creation of the URL of the resource. Section 5.2 of provides a recommendation regarding the generation of the value of this field.

Note: This type can be shared by multiple specifications, which can each define their own instances of root or child elements of this type.

6.2.15 Type: MediaInfo

This type defines media information.

Element	Type	Optional	Description
media	Media	No	type of media information
mediaDirection	MediaDirection	No	direction of media

Table 13: MediaInfo

6.2.16 Type: LanguageString

String with an attribute that signals the language of the contained text.

This type is defined as a string of base type xsd:string with an OPTIONAL instance of the built-in XML attribute xml:lang [W3C-XML11].

6.2.17 Enumeration: Media

This enumeration defines possible values for media.

Enumeration	Description
Audio	Audio media type
Video	Video media type
Chat	Chat media type
Data	Other media type

Table 14: Media

6.2.18 Enumeration: MediaDirection

This enumeration defines possible values for media direction.

Enumeration	Description
In	Incoming
Out	Outgoing
InOut	Bidirectional

Table 15: MediaDirection

6.2.19 Enumeration: CallParticipantStatus

List of the status values associated with a call to a participant.

Enumeration	Description

CallParticipantInitial	The call is being established to a participant
CallParticipantConnected	The participant is active in the call
CallParticipantTerminated	The call to the participant was terminated

Table 16: CallParticipantStatus

6.2.20 Enumeration: CallParticipantTerminationCause

List of the termination causes associated with a call to a participant.

Enumeration	Description
CallParticipantNoAnswer	The participant did not answer the call
CallParticipantBusy	The participant was busy
CallParticipantNotReachable	The participant was not reachable
CallParticipantHangUp	The participant hung up thereby terminating the call for that party
CallParticipantAborted	The call was aborted for the participant (i.e. any other termination cause than hanging up)

Table 17: CallParticipantTerminationCause

6.3 Service and Policy exceptions

In case of errors, additional information in the form of Exceptions MAY be included in the HTTP response.

Exceptions are defined with three data elements.

The first data element 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 element 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.

The third data element 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 element with the first entry mapping to the placeholder %1.

6.3.1 Service exception

The *Service exception* is provided in an XML data type, using the following schema.

```
    <xsd:complexType name="ServiceException">
  <xsd:sequence>
    <xsd:element name="messageId" type="xsd:string"/>
    <xsd:element name="text" type="xsd:string"/>
    <xsd:element maxOccurs="unbounded" minOccurs="0" name="variables" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

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. The ‘SVC’ service exceptions are defined in [ParlayX_Common]

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

6.3.2 Policy exception

The policy exception is provided in an XML data type, using the following schema.

```
    <xsd:complexType name="PolicyException">
  <xsd:sequence>
    <xsd:element name="messageId" type="xsd:string"/>
    <xsd:element name="text" type="xsd:string"/>
    <xsd:element maxOccurs="unbounded" minOccurs="0" name="variables" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

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 *Policy Exception* 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. The ‘POL’ service exceptions are defined in [ParlayX_Common]

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

6.3.3 HTTP Response Codes in response to Notifications

Handling of HTTP response codes sent by the client application, in response to a notification from the server:

1. in case of HTTP 2xx response codes, server assumes the notification has been sent successfully.
2. in case of HTTP response codes other than 2xx, the handling is left to the server implementation. The server MAY support different actions as dictated by a service provider policy (out-of-scope for this specification).

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-ParlayREST_Common-V1_1-20120724-A	24 Jul 2012	Status changed to Approved by TP Ref TP Doc# OMA-TP-2012-0280-INP_ParlayREST_2_0_for_Final_Approval

Appendix B. Static Conformance Requirements

As this specification contains items that are common for the ParlayREST enabler and will not specify any function or interface itself, there are no static conformance requirements and hence this section is intentionally left empty.

Appendix C. Principles for defining REST bindings for Parlay X Web Service APIs (Informative)

[REST_WP] provides generic guidelines and best practices for the specification of REST interfaces in OMA. This informative appendix provides additional guidelines and best practices for the set of ParlayREST APIs.

1. For ParlayREST APIs, the general principles in [REST_WP] section **Error! Reference source not found.** apply. In addition, the following guidelines apply:
    a. As far as possible, when the goal is to use a REST architectural style in transforming APIs previously bound to a different set of protocols, the operations available through REST APIs should provide an equivalent level of functionality as was provided by the original set of APIs, and should use similar data elements (when applicable accordingly with the chosen REST architectural style). In the particular case of ParlayREST, the operations made available through ParlayREST API should provide an equivalent level of functionality to the ParlayX SOAP API subset selected for such transformation, and should use similar data elements (when applicable accordantly with the chosen REST architectural style).
    b. It is recommended that REST binding operations are based on the latest version of 3GPP TS 29 series, release 8. Exceptions should be noted in the specific ParlayREST TS.
    c. The name of the equivalent SOAP binding operation is recommended to be specified in the description of the respective REST binding operation for the better understanding and consistency wherever applicable.
2. For ParlayREST API documentation, the guidelines in [REST_WP] section **Error! Reference source not found.** apply. In addition, for each operation a mapping to the original Parlay X (SOAP) API should be included, where applicable.
3. For error handling, the guidelines in [REST_WP] section **Error! Reference source not found.** apply. In addition, the fault definitions of the REST binding should follow those of the Parlay X SOAP binding. Proprietary extensions might be supported.
4. For MIME encoding and serialization, the guidelines in [REST_WP] section 5.3 apply. In addition, messages with multiple body parts in Parlay X [3GPP 29.199-5] make use of SOAP with attachments (multipart/related MIME type).