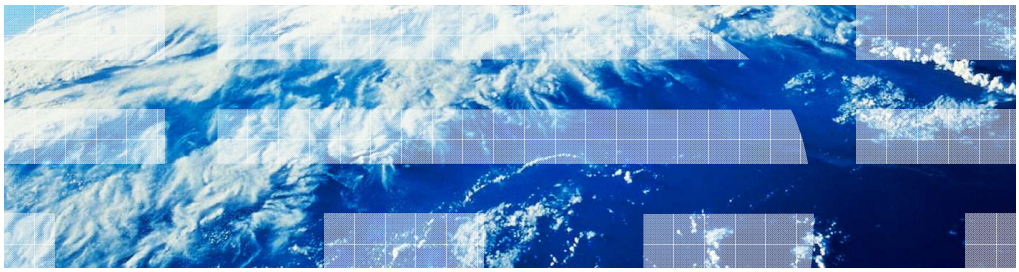
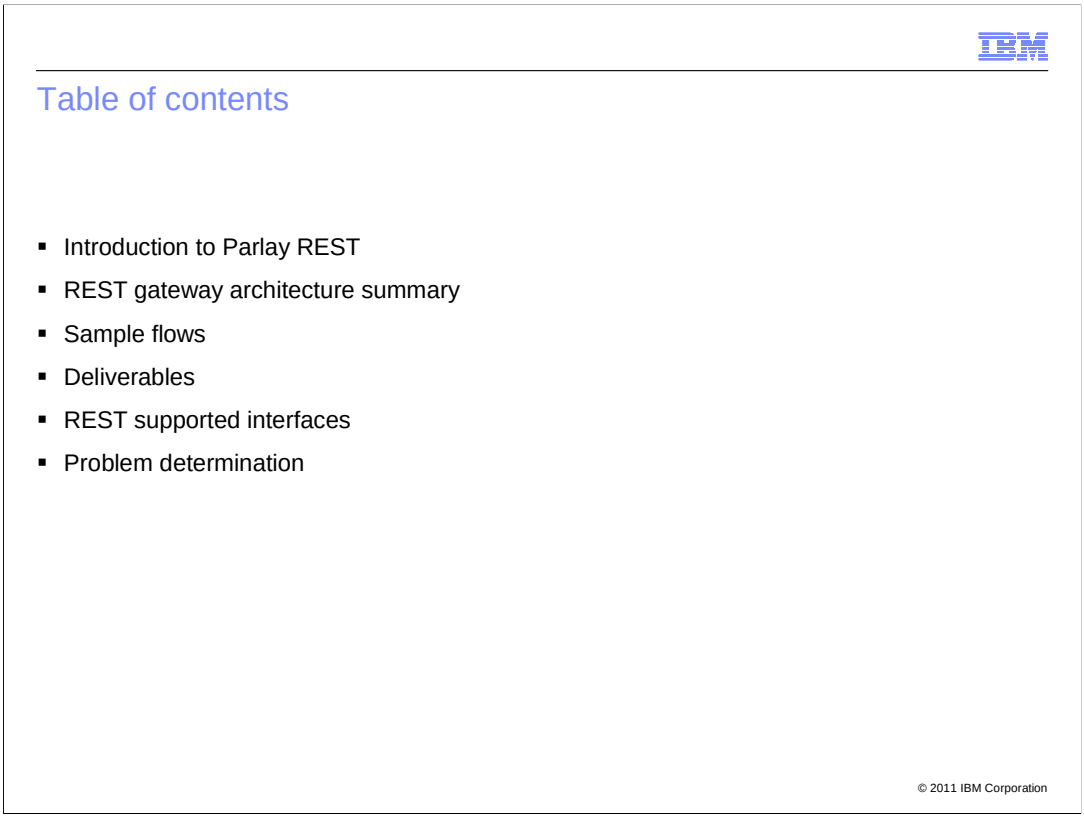
IBM Communication Service Enablers V7.2

REST gateway support in TWSS V7.2 using Parlay REST



This presentation deals with REST Gateway support in Telecom Web Services Server (TWSS) 7.2 using Parlay REST.



Telecom Web Services Server (TWSS) 7.2 release has enhanced REST gateway support. This is provided as part of the implementation based on the Parlay REST specification.

The agenda includes:

- Introduction on Parlay REST to explain REST enabling of the services being exposed
- REST gateway summary and enabling REST support in TWSS 7.2
- Sample flows both for mobile terminated messages and mobile originated messages
- REST supported interfaces
- Problem determination

Guidelines for Parlay REST API specifications

- REST APIs intended for use by the web developer
- Services should be defined as entities/resources and URLs defined accordingly, using nouns not verbs. Messages, Location, Subscribers, Calls, and so on become resources.
- Use of HTTP verbs - POST, GET, PUT, DELETE - for all interfaces, using this mapping:
  - POST maps to Create
  - GET maps to Read
  - PUT maps to Update
  - DELETE maps to Delete
- Detailed specifications: http://www.openmobilealliance.org/Technical/release_program/parlayREST_v1_0.aspx

This slide deals with the guidelines for Parlay REST API specifications.

Parlay REST is a specification provided by OMA (Open Mobile Alliance).

REST is not a technology but a set of principles or an architectural style that makes resources available so that web applications can access and modify them.

Firstly, the REST APIs are intended for use by web developers. It is assumed that the web developers do not have a detailed understanding of the telecom services and hence these specified REST services should be made easy to use, just like any other popular REST services provided on the web.

Secondly, the services should be defined in terms of resources that are addressable as URIs. So the services should be defined as entities and generally these are nouns, not verbs. Messages, locations, subscribers - all these are considered as resources. You can perform create, read, update, and delete operations on these resources. And each of these operations have corresponding HTTP methods associated with it.

For example:

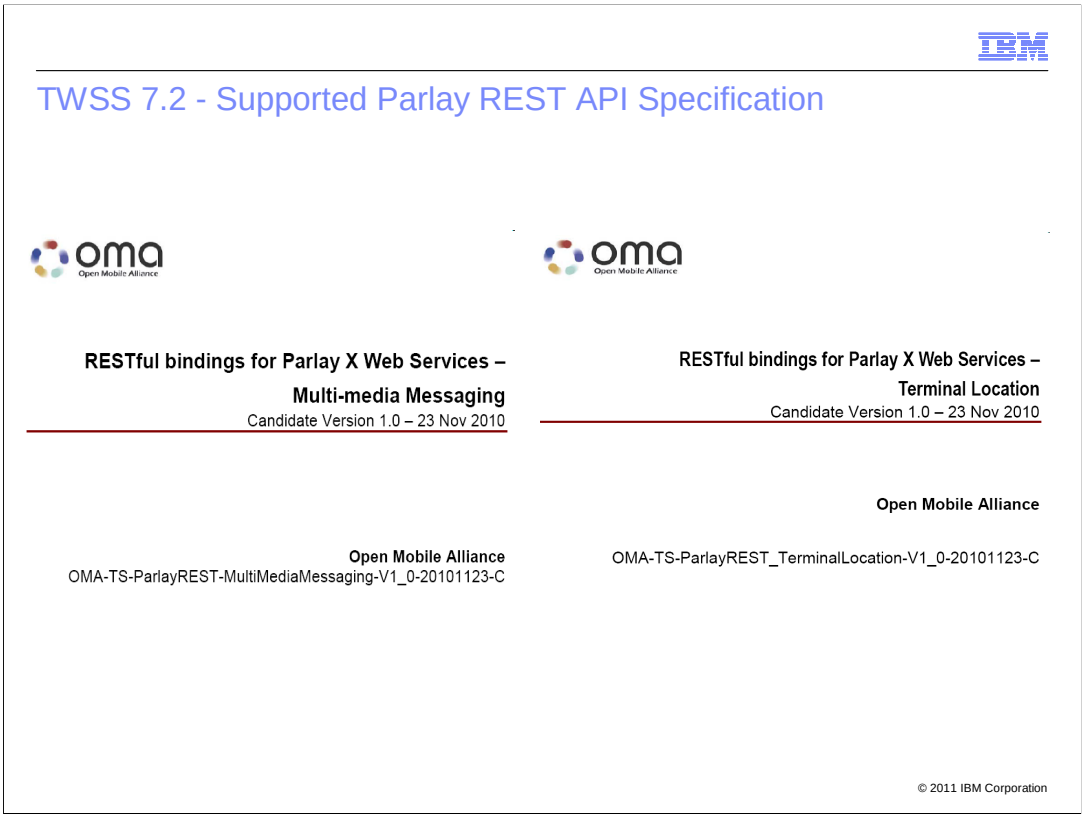
POST maps to Create; GET maps to Read; PUT maps to Update; and DELETE maps to Delete.

There are about 10-12 principles.

The detailed specification guidelines can be downloaded from http://www.openmobilealliance.org/Technical/release_program/parlayREST_v1_0.aspx.

For example: To retrieve the position of a mobile user, the resource required is "location", which is a noun. You need to read the location, which translates to "http get" operation. This resolves into a GET LOCATION operation.

TWSS 7.2 - Supported Parlay REST API Specification



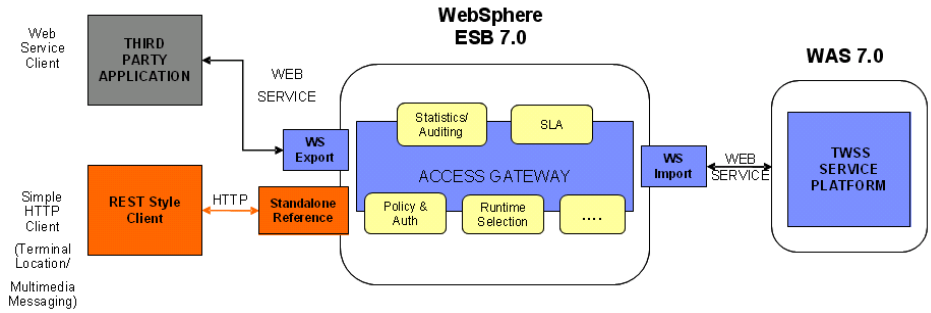
This slide provides screen captures of the two supported operations - Multimedia Messaging and Terminal Location.

The candidate version is 1.0 23rd November 2010.

These Parlay REST standards are specified by OMA standard body, which develops open standards for the mobile phone industry.

It also liaises with the other standard bodies to avoid overlap in specification like 3GPP, IETF, and so on.

TWSS 7.2 – Rest gateway architecture



This slide provides the REST Gateway Architecture for TWSS 7.2 and an overall view of the three major components of the TWSS product - TWSS Access Gateway, SPM (which is not shown), and Service Platform.

The Access Gateway is built on WebSphere® Enterprise Service Bus (ESB) and is the point of entry for all the requests originating from a third party application.

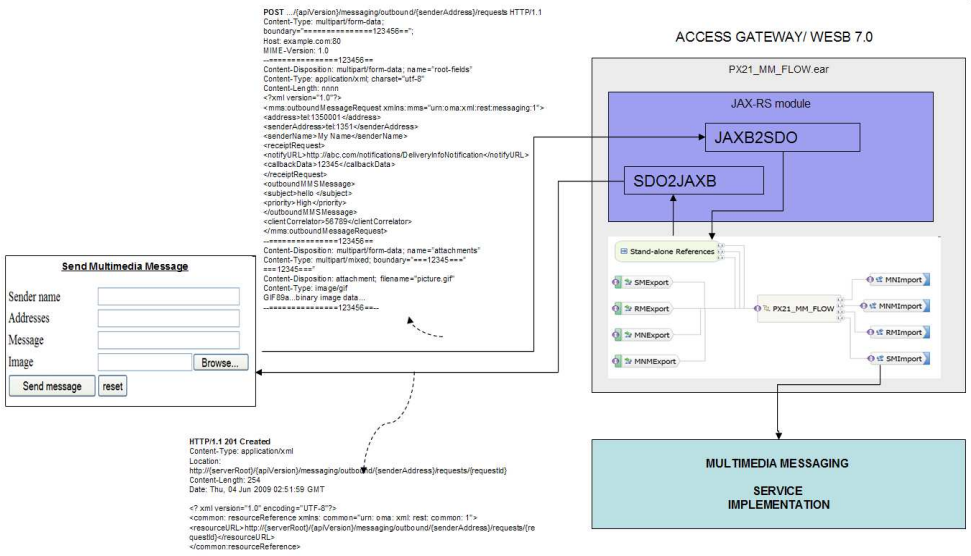
The new services (shown in orange color) are - Terminal Location and Multimedia Messaging.

REST support is built on the Access Gateway component.

In this release, REST support has been extended to Terminal Location and Multimedia Messaging.

The Access Gateway uses a concept called mediation flows and these flows generally remain unchanged.

Parlay REST flow diagram for multimedia messaging (mobile terminated message)



This slide gives a detailed view of the architecture. Although this is the Parlay REST flow diagram for Multimedia Messaging, it applies to Terminal Location as well.

The design uses the libraries defined in the IBM Web 2.0 feature pack, which contains IBM's implementation of the JAX_RS (JSR 311) specification.

JAX_RS is a collection of interfaces and annotations that requires REST application development; and development of RESTful services becomes simplified using JAX_RS.

This diagram shows the flow when an XML REST request from a web page is received. On the right side, you can see the PX21_MM_FLOW.ear, which contains JAX_RS module in blue and the ESB mediation flow module in white.

When an XML request from a web page arrives, it is intercepted by the JAX_RS module and the entire request is captured into a JAXB object.

To get JAXB objects, JAXB classes for the request and response objects are generated from the XML Schema Definitions (XSDs), which are provided along with the Parlay REST specifications.

Depending on the URL, the resource requested, and the HTTP method, a particular method is invoked and this method retrieves the REST parameters from the JAXB objects and maps them to a service data object that the Access Gateway mediation flow understands.

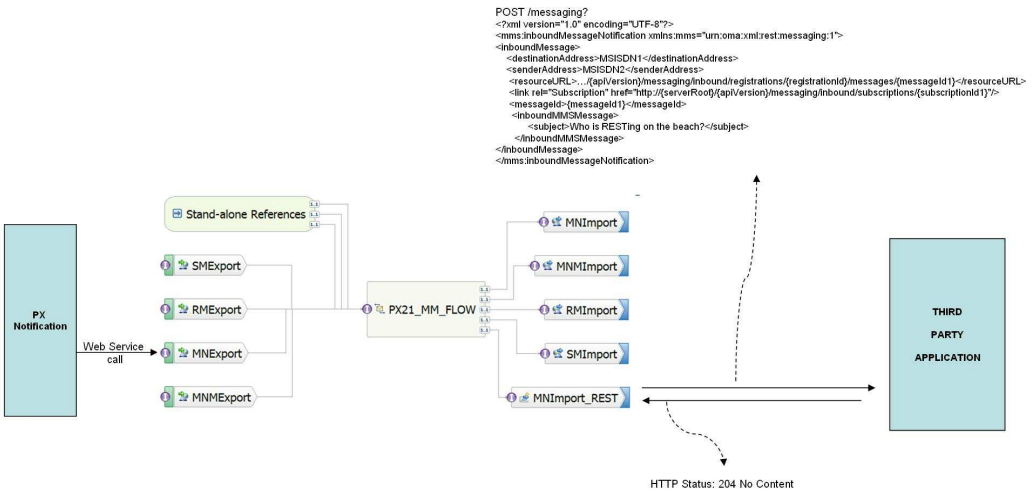
Then it invokes the corresponding operation in the Access Gateway mediation flow through a stand-alone reference. Stand-alone reference is a reference that permits non-SCA (Service Component Architecture) applications to interact with the SCA architecture components. Here the mediation flow is an SCA component.

The flow downstream remains the same as any web service request and the mediation flow gets executed. The service request goes to the back-end Multimedia Messaging service implementation, which responds back with appropriate response and the response arrives back at the response flow of the mediation flow.

Once the response arrives back into the JAX_RS module, the reverse happens. That is, the SDO (Service Data Object) is mapped to the XML with the appropriate HTTP status, which is as per the Parlay REST specification.

For example, the Multimedia Messaging operation is described here. But this entire flow remains the same for the Terminal Location operation too.

Parlay REST flow diagram for multimedia messaging (mobile originated message)



While the previous slide dealt with Mobile Terminated Messages, this slide deals with the Parlay REST flow diagram for Mobile Originated Messages.

When a mobile originated message is received, the Parlay REST client needs to be notified about the message arrival at a callback URI specified by the Third Party Application.

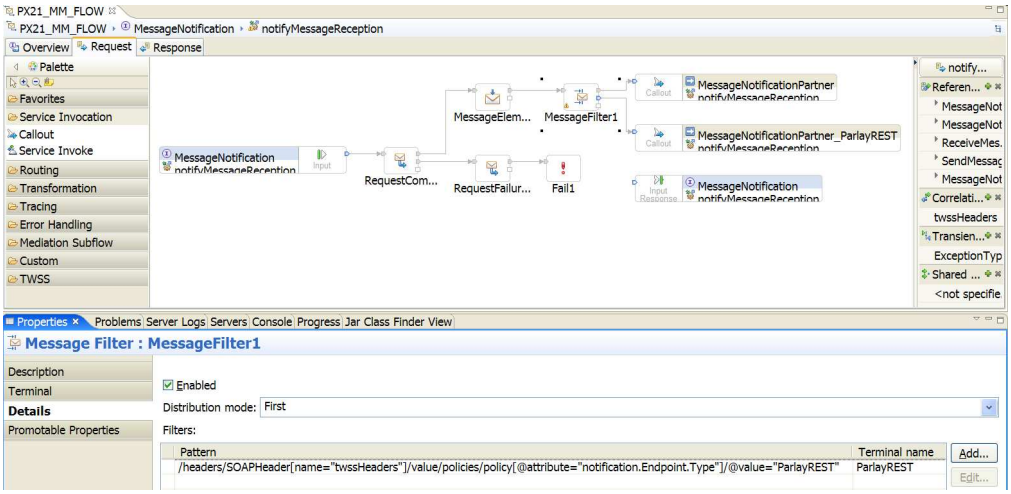
Parlay REST specification provides the message format in which the Third Party Application Callback URI or a 'Catcher' expects the notifications.

The mobile originated message triggers a notification request. The corresponding correlator, requesterID, and Endpoint is fetched and passed as a SOAP header by the PX Notification system to the Access Gateway Notification flow.

The notification message is routed to the dynamic SCA import and the Parlay REST Callback URI is called. There is a handler within the import component, which transforms the Service Message Object (SMO) to the native HTTP Parlay REST format as per the Parlay REST specification.

So the message goes out to the third party application Callback URI and the message is as per the Parlay REST format.

Parlay REST - Mobile originated message (continued)



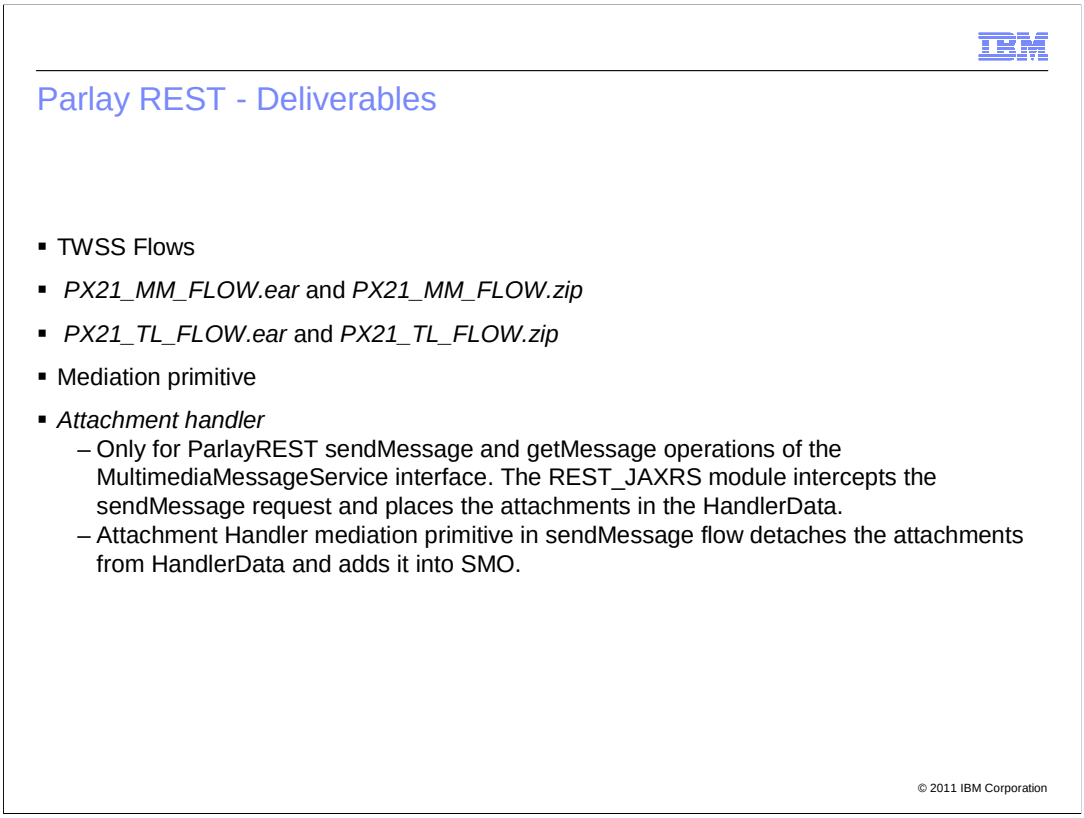
- Policy configuration
  - Policy Name = **notification.Endpoint.Type**
  - Type = String
  - Possible value = 'ParlayREST'

A policy configuration is required for the Mobile Originated flows. Notifications can be received for subscriptions done through:
- web services
- Parlay REST

To distinguish between the two notifications, a policy (notification.Endpoint.Type) needs to be configured so that notifications corresponding to those started from a web service client goes to a web service catcher and the ones for those started from REST client gets directed to REST catcher.

This is entirely policy driven and it is your responsibility to actually ensure that this policy is set.

The Mobile Originated Message passes through the mediation primitives in the flow and eventually reaches a message filter at the end of the request flow. If for that requester, the policy 'notification.Endpoint.Type' has been configured as 'ParlayREST', then the message is routed to the Parlay REST configured endpoint.



For both Terminal Location and Multimedia Messaging flows, the deliverables are as follows:

EAR file contains -

- REST_JAXRS - web module
- WESB - Mediation flow module

An additional mediation primitive, called Attachment Handler, has been introduced in two operations - sendMessage and getMessage for Multimedia Message service only.

NOTE: Multimedia Messaging flow uses JAX-WS bindings. So when a web service request with attachments arrives, the attachments become part of the Service Message Object (SMO).

In the REST equivalent, all the REST messages are intercepted by the JAXRS module first. This JAX-RS module does not have the visibility to the SMO. So it cannot pass on the attachment. The SMO is generated only at runtime. So the attachment is stored by JAXRS module into the HandlerData.

The Attachment Handler mediation primitive residing in the mediation flow reads the contents of the attachments from the HandlerData and populates the attachments element in the SMO with details of the attachments stored.

The rest of the flow remains unchanged.

Parlay REST interfaces are supported in TWSS 7.2 for Multimedia Messaging flow.

The eight supported operations are:

sendMessage: This sends a message to a set of destination addresses.

getMessageDeliveryStatus: The application uses it to retrieve delivery status for each message sent as a result of a previous sendMessage message invocation; after a sendmessage invocation, an ID is received with which the status of the message can be enquired.

NotifyMessageDeliveryReceipt: This provides the Third Party Application with notification of message delivery receipt.

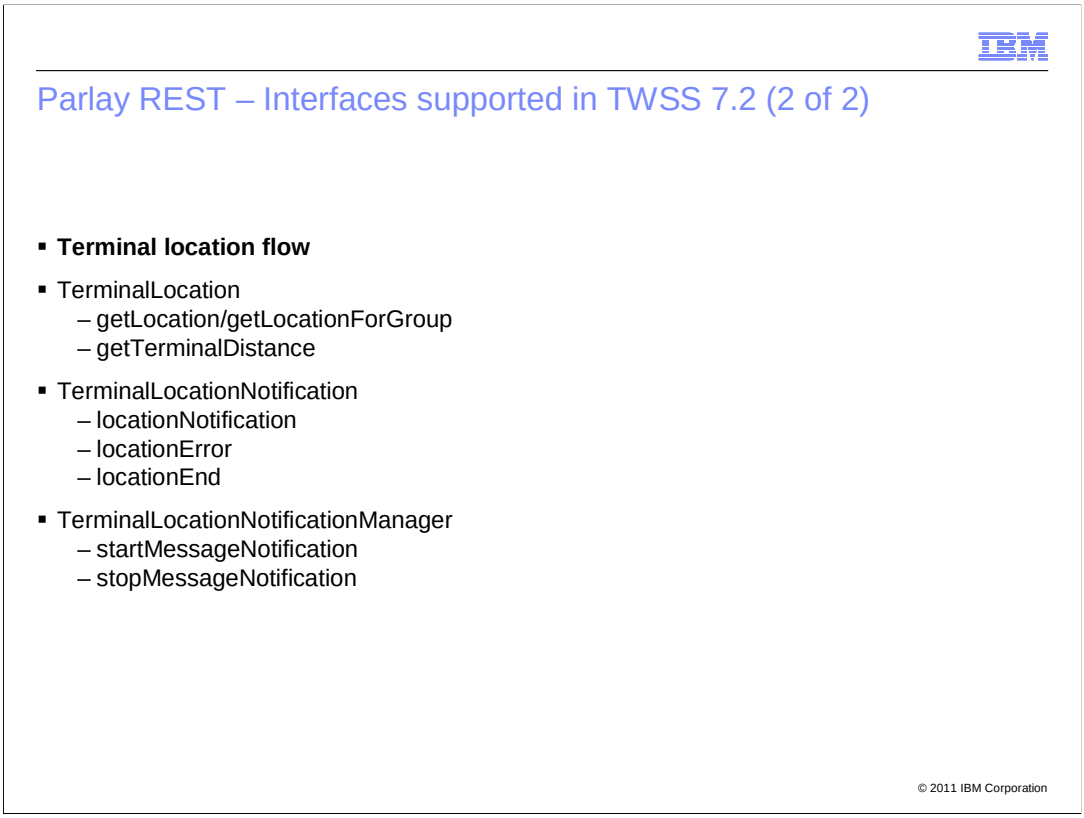
NotifyMessageReception: This is the notification sent to the Third Party Application if a multimedia message fulfils the criteria specified when the multimedia message notification was started.

StartMessageNotification: This starts notifications to the application for a given Message Service activation number and criteria.

stopMessageNotification: This ends a multimedia message notification.

getReceivedMessage: This enables the application to poll for new messages associated with a specific registration ID.

getMessage: This reads or retrieves the whole message; the data is returned as an attachment, as defined in SOAP Messages with Attachments [3], in the return message.



These are the eight operations supported for the Terminal Location flow.

getLocation: This retrieves the location for a single or multiple terminals.

getTerminalDistance: This determines the distance of a terminal from a location

locationNotification: When the location of a monitored device changes, this notification is delivered to the application with the new location information.

locationError: This is sent to the application to indicate that the notification for a terminal, or for the whole notification, is being cancelled by the web service.

locationEnd: This is delivered when the duration or count for notifications have been completed.

In the Notification Manager interface, there are three operations:

startPeriodic: This provides location information for a set of terminals at an application defined interval.

StartGeographicalNotification: This sets up notifications for terminal location events using geographical based definitions.

endNotification: This ends a notification (either type) using this operation.

Mapping Parlay REST to Parlay X operations (Multimedia messaging)

ParlayREST Resource	ParlayREST Method	Parlay X equivalent operation
Inbound messages retrieve and delete using registration	POST	getReceivedMessages
Retrieval and deletion of individual inbound message using registration	POST	getMessage
Inbound message subscriptions	POST	startMessageNotification
Individual inbound message subscription	DELETE	stopMessageNotification
Client notification about inbound message	POST	notifyMessageReception
Outbound message requests	POST	sendMessage
Outbound message delivery status	GET	getMessageDeliveryStatus
Client notification about outbound message delivery status	POST	notifyMessageDeliveryReceipt

For each of the Parlay REST resource there is an equivalent Parlay X operation. The back-end for Multimedia Messaging is based on Parlay X 2.1 specification. And the Parlay REST request is mapped to the Parlay X web service implementation.

The table has the Parlay REST resources listed along with the method that is invoked for each, and the Parlay X equivalent operation for Multimedia Messaging.

Eight operations are supported or exposed through REST APIs for Multimedia Messaging.

The terms “inbound” and “outbound” used in resource names and data structures refer to incoming and outgoing messages from the client perspective.

The term “subscription” refers to the online creation of resources (using requests in this specification).

Mapping Parlay REST to Parlay X operations (Terminal location)

ParlayREST Resource	ParlayREST Method	Parlay X equivalent operation
Terminal location	GET	GetLocation GetLocationForGroup
Periodic location notification subscriptions	POST	StartPeriodicNotification
Individual periodic location notification subscription	DELETE	EndNotification
Area (circle) notification subscriptions	POST	StartGeographicalNotification
Area (circle) individual notification subscription	DELETE	EndNotification
Client notification callback resource	POST POST POST	LocationNotification (Geographical & Periodic) LocationEnd LocationError

Mapping Parlay REST to Parlay X operations for Terminal Location is similar to the Multimedia Messaging mapping. The mapping between the Parlay REST resource for Terminal Location and the equivalent Parlay X operation is shown here.

Nine operations are supported.

The terms “inbound” and “outbound” used in resource names and data structures refer to incoming and outgoing messages from the client perspective

Parlay REST – sendMessage Request

Request

POST path/{apiVersion}/messaging/outbound/{senderAddress}/requests HTTP/1.1
Content-Type: multipart/form-data; boundary="===============123456==";
Host: example.com:80
MIME-Version: 1.0
--===============123456==
Content-Disposition: multipart/form-data; name="root-fields"
Content-Type: application/xml; charset="utf-8"
Content-Length: nnnn
<?xml version="1.0"?>
<mms:outboundMessageRequest xmlns:mms="urn:oma:xml:rest:messaging:1">
<address>tel:7200052011</address>
<senderAddress>tel:7200052013</senderAddress>
<senderName>MyName</senderName>
<receiptRequest>
<notifyURL>http://myserver.com/notifications/DeliveryInfoNotification</notifyURL>
<callbackData>12345</callbackData>
</receiptRequest>
<outboundMMSMessage>
<subject>Test MMS Message</subject>
<priority>High</priority>
</outboundMMSMessage>
<clientCorrelator>052011</clientCorrelator>
</mms:outboundMessageRequest>
--===============123456==
Content-Disposition: multipart/form-data; name="attachments"
Content-Type: multipart/mixed; boundary="===12345==="
===12345==="
Content-Disposition: attachments; filename="picture.gif"
Content-Type: text/plain;
See attached photo
===12345==="
Content-Disposition: attachment; filename="picture.gif"
Content-Type: image/gif
GIF89a...binary image data...
--===============123456==--

Response

HTTP/1.1 201 Created
Content-Type: application/xml
Location:
http://{serverRoot}/{apiVersion}/messaging/outbound/{senderAddress}/requests/{requestId}
Content-Length: 254
Date: Thu, 04 Jun 2009 02:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<mms:outboundMessageRequest
xmlns:mms="urn:oma:xml:rest:messaging:1">
<address>tel:7200052011</address>
<address>tel:7200052012</address>
<senderAddress>tel:1351111999</senderAddress>
<senderName>MyName</senderName>
<outboundMMSMessage>
<subject>Holiday greeings</subject>
</outboundMMSMessage>
<clientCorrelator>12345</clientCorrelator>

<resourceURL>http://{serverRoot}/{apiVersion}/messaging/outbound/{senderAddress}/requests/{requestId}</resourceURL>
</mms:outboundMessageRequest>

This provides an example of a SendMessage request for Parlay REST in the Multimedia Messaging service.

The method used here is POST message and it has multiple attachments that can be sent through the REST request and the response status is HTTP "201 Created".

This means that the resource has been created in terms of REST.

Parlay REST – notifyDeliveryReceipt notification

Request

POST path/notifications/DeliveryInfoNotification/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml; charset=UTF-8
Host: application.example.com:80

```
<?xml version="1.0" encoding="UTF-8"?>
<mms:deliveryInfoNotification xmlns:mms="urn:oma:xml:rest:messaging:1">
        <deliveryInfo>
                <address>tel:1350000001</address>
                <deliveryStatus>DeliveredToTerminal</deliveryStatus>
        </deliveryInfo>
        <deliveryInfo>
                <address>tel:1350000999</address>
                <deliveryStatus>DeliveredToTerminal</deliveryStatus>
        </deliveryInfo>
</mms:deliveryInfoNotification>
```

Response

HTTP/1.1 200 OK
Content-Type: application/xml
Date: Thu, 04 Jun 2009 02:51:59 GMT

All notification requests generally use POST method.

The URL is not important here because it is a Call back URI and is provided by the Third Party Application.

NotifyDeliveryReceipt notification is sent out to the catcher, which is the third party application callback URI.

Parlay REST – getLocation request

Request

GET
path/{apiVersion}/location/queries/location?resFormat=XML&address=tel:7200052011&address=tel:720052012&Tolerance=LowDelay&requestedAccuracy=1000&acceptableAccuracy=1000 HTTP/1.1
Host: myserver1.com:9080

Response

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1234
Date: Mon, 21 Jun 2011 11:20:00 GMT

```
<?xml version="1.0" encoding="UTF-8"?>
<tl:terminalLocationList xmlns:tl="urn:oma:xml:rest:terminallocation:1">
        <terminalLocation>
                <address>tel:720052012</address>
                <locationRetrievalStatus>Retrieved</locationRetrievalStatus>
                <currentLocation>
                        <latitude>-100.86302</latitude>
                        <longitude>50.277306</longitude>
                        <altitude>120.0</altitude>
                        <accuracy>75</accuracy>
                        <timestamp>2011-06-03T00:27:23.000Z</timestamp>
                </currentLocation>
        </terminalLocation>
        <terminalLocation>
                <address>tel:7200052011</address>
                <locationRetrievalStatus>Error</locationRetrievalStatus>
                <errorInformation>
                        <messageId>SVC0001</messageId>
                        <text>A service error occurred. %1 %2</text>
                        <variables>Location information is not available for</variables>
                        <variables>tel:7200052011</variables>
                </errorInformation>
        </terminalLocation>
</tl:terminalLocationList>
```


This shows the example of a getLocation request with multiple addresses.

In this example, the HTTP method is GET and all the parameters are passed through the URI itself. Due to multiple addresses in the URI, the response contains the status of both the addresses.

One of them has been successful and it has retrieved the status - latitude, longitude, and so on; and for the second address, it has sent an error with the details in the error information field.

Parlay REST – locationNotification

Request

```
POST path/notifications/LocationNotification?resFormat=XML HTTP/1.1
Content-Type: application/xml; charset=UTF-8
Host: myServer.com:9080

<?xml version="1.0" encoding="UTF-8"?>
<tl:subscriptionNotification xmlns:tl="urn:oma:xml:rest:terminallocation:1">
        <callbackData>222</callbackData>
        <terminalLocation>
                <address>tel:1720052011</address>
                <locationRetrievalStatus>Retrieved</locationRetrievalStatus>
                <currentLocation>
                        <latitude>-70.86302</latitude>
                        <longitude>21.277306</longitude>
                        <altitude>1020.0</altitude>
                        <accuracy>10</accuracy>
                        <timestamp>2011-05-06T00:27:23.000Z</timestamp>
                </currentLocation>
        </terminalLocation>
        <enteringLeavingCriteria>Entering</enteringLeavingCriteria>
        <isFinalNotification>false</isFinalNotification>
        <link rel="CircleNotificationSubscription"
        href="http://{serverRoot}/{apiVersion}/location/subscriptions/area/circle/{su
bscriptionId}"/>
</tl:subscriptionNotification>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Date: Thu, 04 Jun 2009 02:51:59 GMT
```


This is the locationNotification going out to the Third Party Application callback URI.

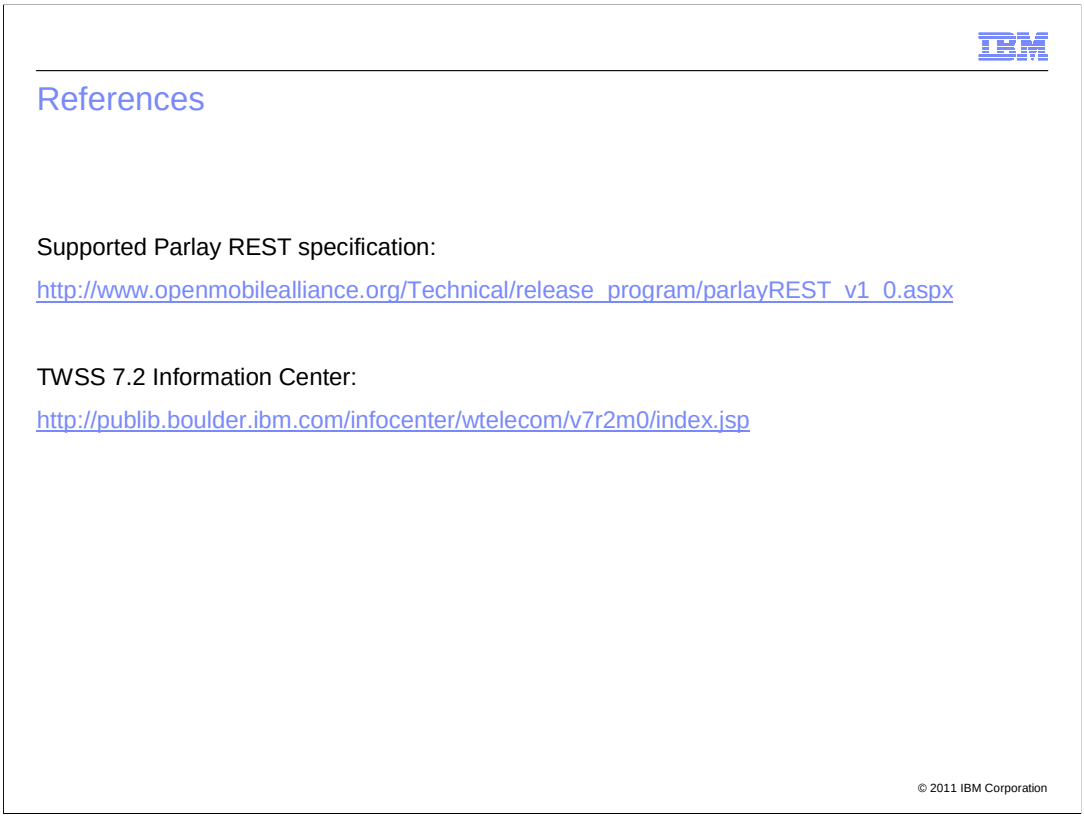
The method used here is POST, which is the standard one used for all the notification operations and the response is HTTP 200 OK.

Parlay REST – Problem determination

- Client test tool
  - Soap UI tool can be used to create REST and JSON requests (SOAP UI: http://www.soapui.org)
    - Recreate problems with customer data
    - Capture request and response messages for debugging
- Log level details
  - com.ibm.soa.esb.parlayrest.common.utils.*=all
  - com.ibm.twss.parlayrest.binding.*=all
  - com.ibm.twss.rest.binding.*=all

The SOAP URI tool can be used to create both REST and JSON requests and it simplifies the creation of a REST request.

For debugging, you can log on to the administration console and navigate to **Troubleshooting- Logging and Tracing** > **server1** > **Change log detail levels**. In the log level details, enable the three trace strings mentioned.



For more information, see:

- REST specification (http://www.openmobilealliance.org/Technical/release_program/parlayREST_v1_0.aspx)
- TWSS 7.2 Information Center (http://publib.boulder.ibm.com/infocenter/wtelecom/v7r2m0/index.jsp).

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_TWSS_7_2_ParlayREST.ppt

This module is also available in PDF format at: ../TWSS_7_2_ParlayREST.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information