# RESTful Network API for Message Broadcast

Candidate Version 1.0 – 13 Sep 2016

**Open Mobile Alliance**

OMA-TS-REST_NetAPI_MsgBCast-V1_0-20160913-C

# Contents

# Figures

# Tables

# 1. Scope

This specification defines a RESTful API for Message Broadcast using HTTP protocol bindings, based on the similar API defined in Parlay X Web Services; part 15, Message Broadcast (release 9) [3GPP TS 29.199-15].

# 2. References

## 2.1 Normative References


**[3GPP TS 23.032]** 3GPP TS 23.032 V12.0.0 "Universal Geographical Area Description (GAD) (Release 12)", 3rd Generation Partnership Project, September 2014, URL:http://www.3gpp.org/

**[3GPP TS 29.199-15]** 3GPP Technical Specification 29.199-15 V9.0.0, "Open Service Access (OSA); Parlay X Web Services; Part 15: Message Broadcast (Release 9)", 3rd Generation Partnership Project, December 2009, URL:http://www.3gpp.org/

**[Autho4API_10]** "Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL:http://www.openmobilealliance.org/

**[OMA ENCap-M2M]** "Exposing Network Capabilities to M2M Requirements", Open Mobile Alliance™, OMA-RD-ENCap_M-V1_0, URL:http://www.openmobilealliance.org

**[REST_NetAPI_ACR]** "RESTful Network API for Anonymous Customer Reference Management ", Open Mobile Alliance™, OMA-TS-REST_NetAPI_ACR-V1_0, URL:http://www.openmobilealliance.org/

**[REST_NetAPI_Common]** "Common definitions for RESTful Network APIs", Open Mobile Alliance™, OMA-TS-REST_NetAPI_Common-V1_0, URL:http://www.openmobilealliance.org/

**[REST_NetAPI_NotificationChannel]** "RESTful Network API for Notification Channel", Open Mobile Alliance™, OMA-TS-REST_NetAPI_NotificationChannel-V1_0, URL:http://www.openmobilealliance.org/

**[REST_SUP_BessageBroadcast]** "XML schema for the RESTful Network API for Message Broadcast", Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_messagebroadcast-V1_0, URL:http://www.openmobilealliance.org/

**[RFC2119]** "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://tools.ietf.org/html/rfc2119.txt

**[RFC3966]** "The tel URI for Telephone Numbers", H.Schulzrinne, December 2004, URL:http://tools.ietf.org/html/rfc3966.txt

**[RFC3986]** "Uniform Resource Identifier (URI): Generic Syntax", R. Fielding et. al, January 2005, URL:http://tools.ietf.org/html/rfc3986.txt

**[RFC7159]** "The JavaScript Object Notation (JSON) Data Interchange Format", T. Bray, Ed., March 2014, URL:http://tools.ietf.org/html/rfc7159.txt

**[RFC7231]** "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, R. Fielding, Ed., J.Raschke, Ed., June 2014, URL:http://tools.ietf.org/html/rfc7231.txt

**[SCRRULES]** "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

**[XMLSchema1]** W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, W3C Recommendation 5 April 2012, URL:http://www.w3.org/TR/xmlschema11-1/

**[XMLSchema2]** W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, W3C Recommendation 5 April 2012, URL:http://www.w3.org/TR/xmlschema11-2/


## 2.2 Informative References


**[OMADICT]** "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

**[REST_WP]** "Guidelines for RESTful Network APIs", Open Mobile Alliance™, OMA-WP-Guidelines_for_RESTful_Network_APIs, URL:http://www.openmobilealliance.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 [OMADICT].

| | |
|---|---|
| **Client-side Notification URL** | An HTTP URL exposed by a client, on which it is capable of receiving notifications and that can be used by the client when subscribing to notifications. |
| **Heavy-weight Resource** | A resource which is identified by a resource URL which is then used by HTTP methods to operate on the entire data structure representing the resource. |
| **Light-weight Resource** | A subordinate resource of a Heavy-weight Resource which is identified by its own resource URL which is then used by HTTP methods to operate on a part of the data structure representing the Heavy-weight Resource. The Light-weight Resource URL can be seen as an extension of the Heavy-weight Resource URL.<br>There could be several levels of Light-weight Resources below the ancestor Heavy-weight Resource, depending on the data structure. |
| **Long Polling** | A variation of the traditional polling technique, where the server does not reply to a request unless a particular event, status or timeout has occurred. Once the server has sent a response, it closes the connection, and typically the client immediately sends a new request. This allows the emulation of an information push from a server to a client. |
| **Notification Channel** | A channel created on the request of the client and used to deliver notifications from a server to a client. The channel is represented as a resource and provides means for the server to post notifications and for the client to receive them via specified delivery mechanisms.<br>For example in the case of Long Polling the channel resource is defined by a pair of URLs. One of the URLs is used by the client as a call-back URL when subscribing for notifications. The other URL is used by the client to retrieve notifications from the Notification Server. |
| **Notification Server** | A server that is capable of creating and maintaining Notification Channels. |
| **Server-side Notification URL** | An HTTP URL exposed by a Notification Server, that identifies a Notification Channel and that can be used by a client when subscribing to notifications. |

## 3.3 Abbreviations

| | |
|---|---|
| **ACR** | Anonymous Customer Reference |
| **API** | Application Programming Interface |
| **HTTP** | HyperText Transfer Protocol |
| **JSON** | JavaScript Object Notation |
| **MIME** | Multipurpose Internet Mail Extensions |
| **OMA** | Open Mobile Alliance |
| **REST** | REpresentational State Transfer |
| **SCR** | Static Conformance Requirements |
| **SIP** | Session Initiation Protocol |

| | |
|---|---|
| **TS** | Technical Specification |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Locator |
| **WP** | White Paper |
| **XML** | eXtensible Markup Language |
| **XSD** | XML Schema Definition |

# 4. Introduction

The Technical Specification of the RESTful Network API for Message Broadcast contains HTTP protocol bindings for the Parlay X Web Services Part 15: Message Broadcast [3GPP TS 29.199-15] specification, using the REST architectural style. The specification provides resource definitions, the HTTP verbs applicable for each of these resources, and the element data structures, as well as support material including flow diagrams and examples using the various supported message body formats (i.e. XML, JSON).

This API is based on an existing SOAP API for Message Broadcast which allows a third-party to make use of Mobile Network broadcast capabilities to reach a group of Devices in a specified geographic area that are registered with that third-party.

A typical use case for this API is the scenario where a third-party alerts or notifies registered devices (with the third-party) about an activity, such as water flooding, accidents etc. See the detailed use cases in OMA Exposing Network Capabilities to M2M [OMA ENCap-M2M].

## 4.1 Version 1.0

The RESTful Network API for Message Broadcast V1.0 is a part of the suite of OMA RESTful Network APIs. The requirements for this API are found in OMA Exposing Network Capabilities to M2M [OMA ENCap-M2M]. While this API is based on [3GPP TS 29.199-15], bug fixes and structural changes to fit that suite, as well as other functional enhancements to meet the requirements [OMA ENCap-M2M] are applied.

Version 1.0 of this specification supports the following operations:

- Send a broadcast message to Devices
- Specify geographic areas which the message should reach
- Specify the time slot and the number of broadcasting
- Check delivery status of the message

While the operations shown below are deferred for future releases

- Create subscriptions for notifications for results of broadcasting
- Delete subscriptions for notifications for results of broadcasting

In addition this specification provides:

- Support for scope values used with authorization framework defined in [Autho4API_10]
- Support for Anonymous Customer Reference (ACR) as an end user identifier
- Support for “acr:auth” as a reserved keyword in an ACR

# 5. Message Broadcast API definition

This section is organized to support a comprehensive understanding of the Message Broadcast API design. It specifies the definition of all resources, definition of all data structures, and definitions of all operations permitted on the specified resources.

Common data types, naming conventions, fault definitions and namespaces are defined in [REST_NetAPI_Common].

The remainder of this document is structured as follows:

Section 5 starts with a diagram representing the resources hierarchy followed by a table listing all the resources (and their URL) used by this API, along with the data structure and the supported HTTP verbs (section 5.1). What follows are the data structures (section 5.2). A sample of typical use cases is included in section 5.3, described as high level flow diagrams.

Section 6 contains detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP methods, and the supported HTTP verbs. For each supported HTTP verb, a description of the functionality is provided, along with an example of a request and an example of a response. For each unsupported HTTP verb, the returned HTTP error status is specified, as well as what should be returned in the Allow header.

All examples in section 6 use XML as the format for the message body. JSON examples are provided in Appendix C.

Section 7 contains fault definition details such as Service Exceptions and Policy Exceptions.

Appendix B provides the Static Conformance Requirements (SCR).

Appendix D lists the Message Broadcast equivalent operation for each supported REST resource and method combination, where applicable.

Appendix E provides a list of all Light-weight Resources, where applicable.

Appendix F defines authorization aspects to control access to the resources defined in this specification.

Note: Throughout this document client and application can be used interchangeably.

## 5.1 Resources Summary

This section summarizes all the resources used by the RESTful Network API for Message Broadcast.

The "apiVersion" URL variable SHALL have the value "v1" to indicate that the API corresponds to this version of the specification. See [REST_NetAPI_Common] which specifies the semantics of this variable.

The figure below visualizes the resource structure defined by this specification. Note that those nodes in the resource tree which have associated HTTP methods defined in this specification are depicted by solid boxes.

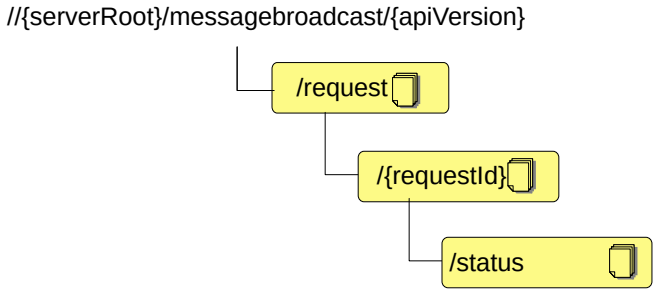


**Figure 1 Resource structure defined by this specification**

The following tables give a detailed overview of the resources defined in this specification, the data type of their representation and the allowed HTTP methods.

**Purpose: To allow an Application (client) to submit broadcast message to a specified area and obtain the status of the message broadcast request.**

| Resource | URL<br>Base URL:<br>http://{serverRoot}/<br>messagebroadcast<br>/{apiVersion} | Data Structures | HTTP verbs | | | |
|---|---|---|---|---|---|---|
| | | | GET | PUT | POST | DELETE |
| Message broadcast request | /request | RequestList<br>(used for GET)<br><br>Request<br>(used for POST) | Retrieve list of broadcast message requests | No | Create a new broadcast message request | No |
| Individual message broadcast request | /request/{requestId} | Request<br>(used for GET and PUT) | Retrieve broadcast message request | Update the broadcast message request | No | Delete a broadcast message request |
| Message broadcast request status | /request/{requestId}/status | Status | Retrieve the status of a message broadcast request | No | No | No |

# 5.2 Data Types

## 5.2.1 XML Namespaces

The XML namespace for the Message Broadcast data types is:

urn:oma:xml:rest:messagebroadcast:1

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

The XML schema for the data structures defined in the section below is given in [REST_SUP_MessageBroadcast].

## 5.2.2 Structures

The subsections of this section define the data structures used in the Message Broadcast API.

Some of the structures can be instantiated as so-called root elements.

For structures that contain elements which describe a user identifier, the statements in section 6 regarding 'tel', 'sip' and 'acr' URI schemes apply.

### 5.2.2.1 Type: RequestList

A type containing list of broadcast message requests.

| Element | Type | Optional | Description |
|---|---|---|---|
| request | Request [0..unbounded] | Choice | Submitted message broadcast request(s) |
| requestReference | common:ResourceReference [0..unbounded] | Choice | Submitted message broadcast request reference(s) (i.e. {requestId}) |
| resourceURL | xsd:anyURI | No | Self referring URL to RequestList. |

A root element named ‘requestList’ of type ‘RequestList’ is allowed in response bodies.

XSD modelling uses a “choice” to select either request or requestReference, but not both of them

### 5.2.2.2 Type: Request

A type containing broadcast message request information.

| Element | Type | Optional | Description |
|---|---|---|---|
| serial | xsd:string | No | An identifier or a tag number of the request provided by the client application |
| broadcastArea | BroadcastArea [1..unbounded] | No | geographical area(s) for which the message is intended to be broadcasted |
| senderName | xsd:string | Yes | If present, it indicates the sender's name of broadcast message, i.e. the string that is displayed on the user's terminal as the originator of the message |

| | | | |
|---|---|---|---|
| charging | Common:ChargingInformation. | Yes | Charge that applies to the message. This consists of description, currency, amount & code. See [REST_NetAPI_Common] for details |
| message | xsd:string | No | Text to be sent in Message Broadcast |
| priority | MessagePriority | Yes | Priority of the message. If not present, the network will assign a priority based on operator's policy. |
| deliveryTime | xsd:dateTime | Yes | If present, it specifies the time to initiate message broadcast in the network. If not present, message is sent immediately |
| totalBroadcasts | xsd:unsignedInt | Yes | The number of broadcasts. If not present, default value is 1. |
| interval | Interval | Yes | The time difference between consecutive broadcasts. It SHALL be present if totalBroadcasts > 1 otherwise it is ignored by the server |
| resourceURL | xsd:anyURI | Yes | Self referring URL.<br><br>The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests. |

A root element named 'request' of type 'Request' is allowed in request and/or response bodies.

### 5.2.2.3 Type: BroadcastArea

A type containing broadcast area information.

| Element | Type | Optional | Description |
|---|---|---|---|
| unionElement | AreaType | No | Type of geographical area (e.g. Circle, Polygon). |
| alias | xsd:string | Yes | An alias name of a geographical area. The alias name shall be understood and translated by network. In POST requests this element SHALL be present only if areaType is "Alias" otherwise it SHALL be ignored by the server. |
| circle | Circle | Yes | Circle shaped broadcast area. In POST requests this element SHALL be present only if areaType is "Circle" otherwise it SHALL be ignored by the server. |
| polygon | Polygon | Yes | Polygon shaped broadcast area. In POST requests this element SHALL be present only if areaType is "Polygon" otherwise it SHALL be ignored by the server. |

### 5.2.2.4 Type: LocationPoint

A type containing geographical points information e.g. latitude and longitude.

| Element | Type | Optional | Description |
|---|---|---|---|
| latitude | xsd:float | No | latitude value of a location |
| longitude | xsd:float | No | longitude value of a location |

### 5.2.2.5 Type: Circle

A type containing circle area defining centre and radius information for the shape.

| Element | Type | Optional | Description |
|---|---|---|---|
| centre | LocationPoint | No | The centre point of circle |
| radius | xsd:float | No | radius of circle (in meters) |

### 5.2.2.6 Type: Polygon

A type containing polygon area defining vertices for an arbitrary shape.

| Element | Type | Optional | Description |
|---|---|---|---|
| locationPoints | LocationPoint [3..15] | No | Set of location points to make a polygon. See also clause 5.4, 7.3.4 of 3GPP TS 23.032 [3GPP TS 23.032]. |

### 5.2.2.7 Type: Status

A type containing list of broadcast status for corresponding broadcast request messages.

| Element | Type | Optional | Description |
|---|---|---|---|
| link | common:Link | Yes | Link reference for the resource representing the request (i.e., {requestId}) |
| statusResults | StatusData [1..unbounded] | No | Set of results for the request. It provides the broadcast status for each area with several supplementary data like the number of broadcast, success rate, broadcast end time. Possible status values are:<br>– MessageWaiting<br>– Broadcasting<br>– Broadcasted<br>– BroadcastImpossible<br>– BroadcastUnknown |
| resourceURL | xsd:anyURI | No | self referring URL. |

A root element named ‘status of type ‘Status is allowed in response bodies.

### 5.2.2.8 Type: StatusData

A type containing area and its status.

| Element | Type | Optional | Description |
|---|---|---|---|
| area | BroadcastArea | No | A broadcast area |

| | | | |
|---|---|---|---|
| reportStatus | RetrievalStatus | No | Status of retrieval for this broadcast area. See [REST_NetAPI_Common] for details. |
| currentStatus | StatusInfo | Yes | Broadcast status of this area. It SHALL be provided by the Server if reportStatus=Retrieved. |
| errorInformation | common:ServiceError | Yes | If reportStatus = Error, this is the reason for the error. See [REST_NetAPI_Common] for details. |

#### 5.2.2.9 Type: StatusInfo

A type containing broadcast status information of an area.

| Element | Type | Optional | Description |
|---|---|---|---|
| status | BroadcastStatus | No | Broadcast status of this area. |
| numberOfBroadcasts | xsd:unsignedInt | Yes | The number of broadcasts successfully sent out. This is optional and present only if status is either Broadcasting or Broadcasted. |
| successRate | xsd:unsignedInt | Yes | Successful delivery rate expressed as a percentage. This is optional and present only if status is either Broadcasting or Broadcasted. |
| broadcastEndTime | xsd:dateTime | Yes | Completed time of broadcast. This is optional and present only if status is Broadcasted. |

### 5.2.3 Enumerations

The subsections of this section define the enumerations used in the Message Broadcast API.

#### 5.2.3.1 Enumeration: BroadcastStatus

An enumeration defining broadcast delivery status values.

| Enumeration | Description |
|---|---|
| MessageWaiting | The message is still queued and not delivered to the network yet. Broadcasting has not commenced. |
| Broadcasting | Broadcasting is initiated and the network is still attempting to deliver messages: i.e., as many times as requested in the total broadcasts. |
| Broadcasted | A final state that indicates broadcast requests were successfully delivered to network: i.e., as many times as requested. |
| BroadcastImpossible | Delivery of broadcast message is impossible. Reasons include: 'out of network coverage', 'network overloads', 'expiry of valid period'. |
| BroadcastUnknown | Delivery status unknown: e.g., delivery requested but no response. |
| BroadcastNotificationNotSupported | Unable to provide broadcast delivery receipt notification. |

#### 5.2.3.2 Enumeration: AreaType

An enumeration defining the types of area that may be used to define broadcast area for message broadcast request.

| Enumeration | Description |
|---|---|
| Alias | Alias name shared by both application and network |
| Circle | Area represented as a circle shape |
| Polygon | Area represented as a polygon shape |

#### 5.2.3.3 Enumeration: MessagePriority

An enumeration defining delivery priority values for the message broadcast request.

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

### 5.2.4 Values of the Link "rel" attribute

The "rel" attribute of the Link element is a free string set by the server implementation, to indicate a relationship between the current resource and an external resource. The following are possible strings (list is non-exhaustive, and can be extended):

- RequestReference

These values indicate the kind of resource that the link points to.

## 5.3 Sequence Diagrams

The following subsections describe the resources, methods and steps involved in typical scenarios.

In a sequence diagram, a step which involves delivering a notification is labeled with "POST or NOTIFY", where "POST" refers to delivery via the HTTP POST method, and "NOTIFY" refers to delivery using the Notification Channel [REST_NetAPI_NotificationChannel].

### 5.3.1 Requesting to send a broadcast message in specified geographic areas

This figure below shows a scenario for requesting the server to send a broadcast message in specified geographic areas.

The resources:

- To request sending broadcast message, create resource under **http://{serverRoot}/messagebroadcast/{apiVersion}/request**
- To retrieve the message broadcast status under requestId, read resource under **http://{serverRoot}/messagebroadcast/{apiVersion}/request/{requestId}/status**

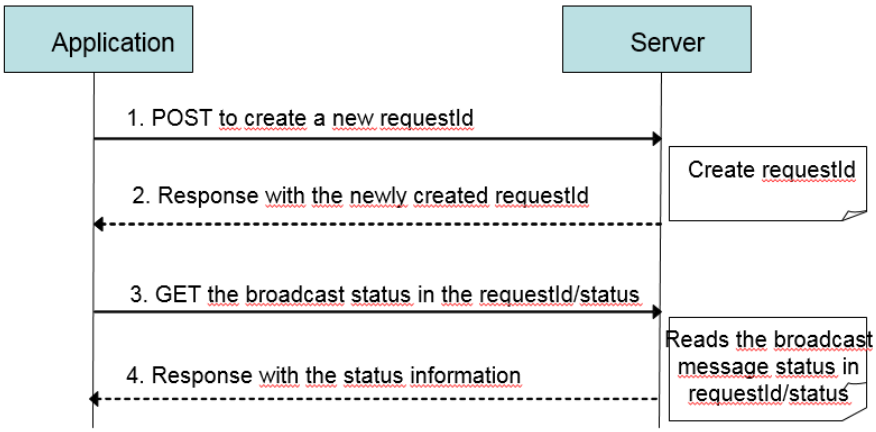


**Figure 2 Flow for sending a broadcast message**

Outline of the flows:

1. An application POSTs a request to the server. The request contains a "Request" data structure (1.e. the representation of {requestId} resource which is to be created by the server.
2. The application receives the newly created resource URL containing the {requestID}. This response is an acknowledgement of the request and does not have the result of broadcasting. The server, at the same time, starts sending the broadcast message with specified conditions and allocates a new {requestId}/status resource representing the status of broadcasting.
3. The application requests the delivery status of the sent broadcast message using GET method on requestId/status
4. The server responds with the delivery status information.

### 5.3.2 Retrieve message broadcast delivery status

This figure below shows a scenario retrieving delivery status of a broadcast message send to a specific geographic area defined by an application.

The resources:

- To retrieve the issued list of message broadcasts, read resource under **http://{serverRoot}/messagebroadcast/{apiVersion}/request**
- To retrieve the message broadcast status under requestId, read resource under **http://{serverRoot}/messagebroadcast/{apiVersion}/request/{requestId}/status**

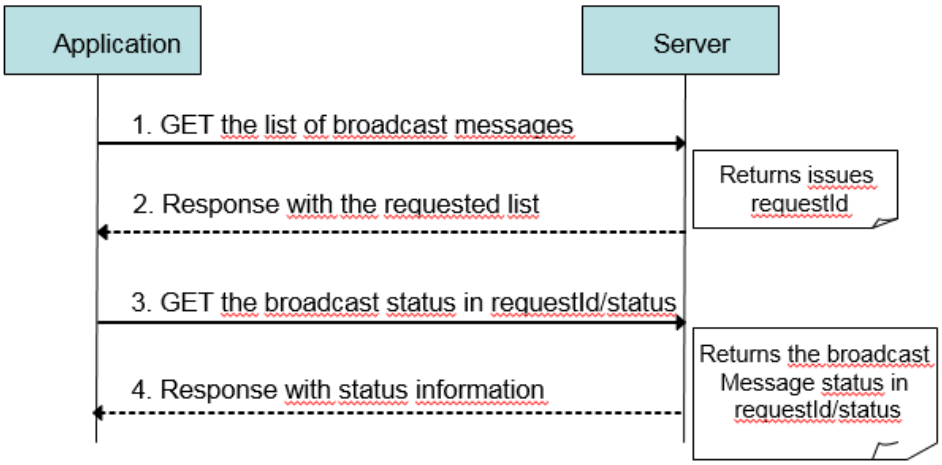


**Figure 3 Flow for retrieving message delivery status**

Outline of the flows:

1. An application requests the issued list of broadcast messages using GET method
2. The server responds with the list broadcast messages and their requestIds.
3. The application requests the delivery status of the sent broadcast message using GET method on requestId/status
4. The server returns relevant status information.

### 5.3.3 Deleting a broadcast message

This figure below shows a scenario for deleting a broadcast message send to a specific geographic area defined by an application.

The resources:

– To delete a message broadcast data under requestId, delete resource under

**http://{serverRoot}/messagebroadcast/{apiVersion}/request/{requestId}**

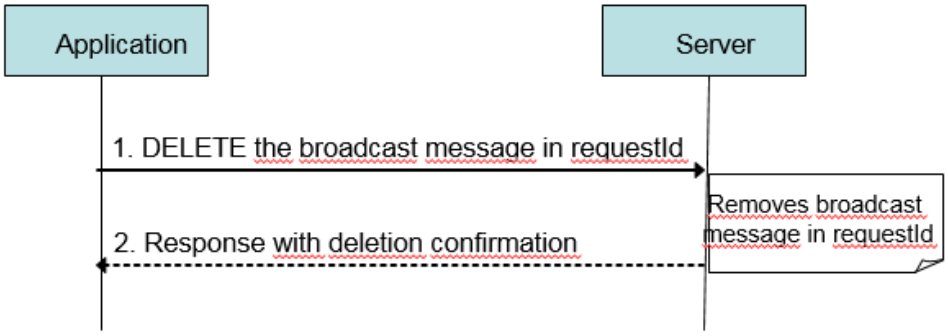


**Figure 4 Flow for deleting a broadcast message**

Outline of the flows:

1. The application decides to remove one of the broadcast messages in the list by using DELETE method on the resource
2. The server returns a response with deletion confirmation.

# 6. Detailed specification of the resources

The following applies to all resources defined in this specification regardless of the representation format (i.e. XML, JSON):

- Reserved characters in URL variables (parts of a URL denoted below by a name in curly brackets) MUST be percent-encoded according to [RFC3986]. Note that this always applies, no matter whether the URL is used as a Request URL or inside the representation of a resource (such as in “resourceURL” and “link” elements).
- If a user identifier (e.g. address, participantAddress, etc.) of type anyURI is in the form of an MSISDN, it MUST be defined as a global number according to [RFC3966] (e.g. tel:+19585550100). The use of characters other than digits and the leading “+” sign SHOULD be avoided in order to ensure uniqueness of the resource URL. This applies regardless of whether the user identifier appears in a URL variable or in a parameter in the body of an HTTP message.
- If an equipment identifier of type anyURI is in the form of a SIP URI, it MUST be defined according to [RFC3261].
- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of an Anonymous Customer Reference (ACR), it MUST be defined according to [REST_NetAPI_ACR], i.e. it MUST include the protocol prefix 'acr:' followed by the ACR.
    - The ACR ‘auth’ is a supported reserved keyword, and MUST NOT be assigned as an ACR to any particular end user. See F.1.2 for details regarding the use of this reserved keyword.
- For requests and responses that have a body, the following applies: in the requests received, the server SHALL support JSON and XML encoding of the parameters in the body. The Server SHALL return either JSON or XML encoded parameters in the response body, according to the result of the content type negotiation as specified in [REST_NetAPI_Common]. In notifications to the Client, the server SHALL use either XML or JSON encoding, depending on which format the client has specified in the related subscription. The generation and handling of the JSON representations SHALL follow the rules for JSON encoding in HTTP Requests/Responses as specified in [REST_NetAPI_Common].

## 6.1 Resource: Message broadcast requests

The resource used is:

**http://{serverRoot}/messagebroadcast/{apiVersion}/request**

This resource is used for retrieving an already issued message broadcast request list and for submitting a new broadcast message.

### 6.1.1 Request URL variables

The following request URL variables are common for all HTTP methods:

| Name | Description |
|---|---|
| serverRoot | Server base url: hostname+port+base path. Port and base path are OPTIONAL.<br>Example: example.com/exampleAPI |
| apiVersion | Version of the API client wants to use. The value of this variable is defined in section 5.1 |

See section 6 for a statement on the escaping of reserved characters in URL variables.

### 6.1.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Message Broadcast API, see section 7.

## 6.1.3 GET

This operation is used for retrieving a list of already submitted broadcast messages.

### 6.1.3.1 Example 1: Retrieving a list of broadcast message requests (Informative)

In this example an application is retrieving the list of message broadcast requests that have been submitted.

#### 6.1.3.1.1 Request

```
GET /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/xml
```

#### 6.1.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:requestList xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
 <request>
  <serial>A00001EF</serial>
  <broadcastArea>
   <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5573</latitude>
     <longitude>-0.3930</longitude>
    </centre>
    <radius>2000</radius>
   </circle>
  </broadcastArea>
  <broadcastArea>
   <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5758</latitude>
     <longitude>-0.4212</longitude>
    </centre>
    <radius>2000</radius>
   </circle>
  </broadcastArea>
  <senderName>South Ruislip Traffic Police</senderName>
  <charging>
   <description>Subscription to emergency messaging service</description>
   <currency>GBP</currency>
   <amount>200</amount>
   <code>E24</code>
  </charging>
  <message>Major Traffic Accident at the Polish War Memorial</message>
  <priority>High</priority>
  <deliveryTime>2016-03-26T18:00:00</deliveryTime>
```

```
   <totalBroadcasts>15</totalBroadcasts>
   <interval>7200</interval>
   <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
  </request>
  <request>
   <serial>86066117</serial>
   <broadcastArea>
    <unionElement>Circle</unionElement>
    <circle>
     <centre>
      <latitude>51.6054</latitude>
      <longitude>-0.1222</longitude>
     </centre>
     <radius>2000</radius>
    </circle>
   </broadcastArea>
   <senderName>Harringay Fire Brigade</senderName>
   <message>Building on fire</message>
   <priority>High</priority>
   <deliveryTime>2016-03-26T14:00:00</deliveryTime>
   <totalBroadcasts>10</totalBroadcasts>
   <interval>7200</interval>
   <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0002</resourceURL>
  </request>
  <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request</resourceURL>
</mb:requestList>
```

### 6.1.3.2 Example 2: Retrieving a list of broadcast message requests, response with a list of resource references (Informative)

In this example an application is retrieving the list of message broadcast requests. The response is a list of resource references (i.e. list of {requestID}).

#### 6.1.3.2.1 Request

```
GET /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/xml
```

#### 6.1.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:requestList xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
  <requestReference>
    <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
  </requestReference >
  <requestReference>
    <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0002</resourceURL>
  </requestReference>
<resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request</resourceURL>
</mb:requestList>
```

## 6.1.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: [GET/POST]' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

## 6.1.5 POST

This operation is used for creating a new broadcast message request.

### 6.1.5.1 Example 1: Requesting to send a broadcast message in specified geographic areas (Informative)

In this example an application requests the server to send a broadcast message in specified geographic areas.

#### 6.1.5.1.1 Request

```
POST /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?>
<mb:request xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
 <serial>A00001EF</serial>
 <broadcastArea>
    <unionElement>Circle</unionElement>
   <circle>
    <centre>
      <latitude>51.5573</latitude>
      <longitude>-0.3930</longitude>
    </centre>
    <radius>2000</radius>
  </circle>
 </broadcastArea>
 <broadcastArea>
  <unionElement>Circle</unionElement>
  <circle>
    <centre>
      <latitude>51.5758</latitude>
      <longitude>-0.4212</longitude>
    </centre>
    <radius>2000</radius>
  </circle>
 </broadcastArea>
 <senderName>South Ruislip Traffic Police</senderName>
 <charging>
  <description>Subscription to emergency messaging service</description>
  <currency>GBP</currency>
  <amount>200</amount>
  <code>E24</code>
 </charging>
 <message>Major Traffic Accident at the Polish War Memorial</message>
 <priority>High</priority>
 <deliveryTime>2016-03-26T18:00:00-07:00</deliveryTime>
 <totalBroadcasts>15</totalBroadcasts>
 <interval>7200</interval>
```

```
</mb:request>
```

#### 6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Date: Mon, 07 Mar 2016 15:17:02 GMT
Location: http://example.com/exampleAPI/messagebroadcast/v1/request/b0001
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:request xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
 <serial>A00001EF</serial>
 <broadcastArea>
  <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5573</latitude>
     <longitude>-0.3930</longitude>
    </centre>
    <radius>2000</radius>
   </circle>
 </broadcastArea>
 <broadcastArea>
  <unionElement>Circle</unionElement>
  <circle>
   <centre>
    <latitude>51.5758</latitude>
    <longitude>-0.4212</longitude>
   </centre>
   <radius>2000</radius>
  </circle>
 </broadcastArea>
 <senderName>South Ruislip Traffic Police</senderName>
 <charging>
  <description>Subscription to emergency messaging service</description>
  <currency>GBP</currency>
  <amount>200</amount>
  <code>E24</code>
 </charging>
 <message>Major Traffic Accident at the Polish War Memorial</message>
 <priority>High</priority>
 <deliveryTime>2016-03-26T18:00:00-07:00</deliveryTime>
 <totalBroadcasts>15</totalBroadcasts>
 <interval>7200</interval>
 <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
</mb:request>
```

## 6.1.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: [GET/POST]’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

# 6.2 Resource: Individual message broadcast request

The resource used is:

**http://{serverRoot}/messagerequest/{apiVersion}/request/{requestId}**

This resource is used for retrieving, updating and deleting an individual message broadcast request.

## 6.2.1 Request URL variables

The following request URL variables are common for all HTTP methods:

| Name | Description |
|---|---|
| serverRoot | Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI |
| apiVersion | Version of the API client wants to use. The value of this variable is defined in section 5.1 |
| requestId | Identifier of a particular message broadcast request |

## 6.2.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Message Broadcast API, see section 7.

## 6.2.3 GET

This operation is used to retrieve the message broadcast request related to a particular requestId.

### 6.2.3.1 Example: Retrieve a submitted message broadcast request (Informative)

In this example, a query is initiated by an application to retrieve the message broadcast request related to a particular requestId residing in the network server.

#### 6.2.3.1.1 Request

```
GET /exampleAPI/messagebroadcast/v1/request/b0001 HTTP/1.1
Host: example.com
Accept: application/xml
```

#### 6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:request xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
  <serial>A00001EF</serial>
  <broadcastArea>
    <unionElement>Circle</unionElement>
   <circle>
     <centre>
```

```
      <latitude>51.5573</latitude>
      <longitude>-0.3930</longitude>
    </centre>
    <radius>2000</radius>
  </circle>
 </broadcastArea>
 <broadcastArea>
  <unionElement>Circle</unionElement>
  <circle>
    <centre>
      <latitude>51.5758</latitude>
      <longitude>-0.4212</longitude>
    </centre>
    <radius>2000</radius>
  </circle>
 </broadcastArea>
 <senderName>South Ruislip Traffic Police</senderName>
 <charging>
  <description>Subscription to emergency messaging service</description>
  <currency>GBP</currency>
  <amount>200</amount>
  <code>E24</code>
 </charging>
 <message>Major Traffic Accident at the Polish War Memorial</message>
 <priority>High</priority>
 <deliveryTime>2016-03-26T18:00:00</deliveryTime>
 <totalBroadcasts>15</totalBroadcasts>
 <interval>7200</interval>
 <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
</mb:request>
```

## 6.2.4 PUT

This operation is used to update the message broadcast request related to a particular requestId.

### 6.2.4.1 Example: Update submitted message broadcast request (Informative)

In this example, a request is initiated by an application to update an existing submitted broadcast message.

#### 6.2.4.1.1 Request

```
PUT /exampleAPI/ messagebroadcast/v1/request/b0001 HTTP/1.1
Host: example.com
Accept: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?>
<mb:request xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
 <serial>A00001EF</serial>
 <broadcastArea>
   <unionElement>Circle</unionElement>
  <circle>
    <centre>
      <latitude>51.5573</latitude>
      <longitude>-0.3930</longitude>
```

```
    </centre>
    <radius>3000</radius>
   </circle>
  </broadcastArea>
  <broadcastArea>
   <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5758</latitude>
     <longitude>-0.4212</longitude>
    </centre>
    <radius>2000</radius>
   </circle>
  </broadcastArea>
  <senderName>South Ruislip Traffic Police</senderName>
  <charging>
   <description>Subscription to emergency messaging service</description> <!-- description string -->
   <currency>GBP</currency>
   <amount>200</amount>
   <code>E24</code>
  </charging>
  <message>Major Traffic Accident at the Polish War Memorial</message>
  <priority>High</priority>
  <deliveryTime>2016-03-26T18:00:00-07:00</deliveryTime>
  <totalBroadcasts>15</totalBroadcasts>
  <interval>7200</interval>
  <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
</mb:request>
```

#### 6.2.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:request xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
  <serial>A00001EF</serial>
  <broadcastArea>
    <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5573</latitude>
     <longitude>-0.3930</longitude>
    </centre>
    <radius>3000</radius>
   </circle>
  </broadcastArea>
  <broadcastArea>
   <unionElement>Circle</unionElement>
   <circle>
    <centre>
     <latitude>51.5758</latitude>
     <longitude>-0.4212</longitude>
    </centre>
```

```
    <radius>2000</radius>
   </circle>
  </broadcastArea>
  <senderName>South Ruislip Traffic Police</senderName>
  <charging>
   <description>Subscription to emergency messaging service</description>
   <currency>GBP</currency>
   <amount>200</amount>
   <code>E24</code>
  </charging>
  <message>Major Traffic Accident at the Polish War Memorial</message>
  <priority>High</priority>
  <deliveryTime>2016-03-26T18:00:00-07:00</deliveryTime>
  <totalBroadcasts>15</totalBroadcasts>
  <interval>7200</interval>
  <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001</resourceURL>
</mb:request>
```

### 6.2.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: [GET/PUT/DELETE] field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

### 6.2.6 DELETE

This operation is used to delete a message broadcast request if required.

#### 6.2.6.1 Example: Delete submitted message broadcast request (Informative)

In this example, a request is initiated by an application to delete an existing submitted broadcast message.

##### 6.2.6.1.1 Request

```
DELETE/exampleAPI/messagebroadcast/v1/request/b0001 HTTP/1.1
Accept: application/xml
Host: example.com
```

##### 6.2.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Date: Tue, 7 Mar 2016 10:50:00 GMT
```

## 6.3 Resource: Message Broadcast request status

The resource used is:

**http://{serverRoot}/messagebroadcast/{apiVersion}/request/{requestId}/status**

This resource is used for retrieving the status of a message broadcast request.

### 6.3.1 Request URL variables

The following request URL variables are common for all HTTP methods:

| Name | Description |
|---|---|

| | |
|---|---|
| serverRoot | Server base url: hostname+port+base path. Port and base path are OPTIONAL.<br>Example: example.com/exampleAPI |
| apiVersion | Version of the API client wants to use. The value of this variable is defined in section 5.1 |
| requestId | Identifier of a particular message broadcast request |

See section 6 for a statement on the escaping of reserved characters in URL variables.

## 6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Message Broadcast API, see section 7.

## 6.3.3 GET

This operation is used for retrieving the status of a message broadcast request of a particular requestId.

### 6.3.3.1 Example: Retrieve the message broadcast request status (Informative)

In this example, a query is initiated by an application to retrieve the status of a message broadcast request.

#### 6.3.3.1.1 Request

```
GET /exampleAPI/messagebroadcast/v1/request/b0001/status HTTP/1.1
Accept: application/xml
Host: example.com
```

#### 6.3.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 7 Mar 2016 10:50:00 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<mb:status xmlns:mb="urn:oma:xml:rest:netapi:messagebroadcast:1">
  <link rel="RequestReference" href="http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"/>
  <statusResults>
    <area>
      <unionElement>Circle</unionElement>
      <circle>
        <centre>
          <latitude>51.5573</latitude>
          <longitude>-0.3930</longitude>
        </centre>
        <radius>3000</radius>
      </circle>
    </area>
    <reportStatus>Retrieved</reportStatus>
    <currentStatus>
      <status>Broadcasted</status>
      <numberOfBroadcasts>15</numberOfBroadcasts>
      <successRate>100</successRate>
      <broadcastEndTime>2016-03-26T21:32:52-00:00</broadcastEndTime>
    </currentStatus>
```

```
  </statusResults>
  <statusResults>
    <area>
      <unionElement>Circle</unionElement>
      <circle>
        <centre>
          <latitude>51.5758</latitude>
          <longitude>-0.4212</longitude>
        </centre>
        <radius>2000</radius>
      </circle>
    </area>
    <reportStatus>Retrieved</reportStatus>
    <currentStatus>
      <status>BroadcastImpossible</status>
      <numberOfBroadcasts>0</numberOfBroadcasts>
      <successRate>0</successRate>
      <broadcastEndTime>2016-03-26T21:32:52-00:00</broadcastEndTime>
    </currentStatus>
    <errorInformation>
       <messageId>SVC0300</messageId>
        <text>Broadcast Area not supported</text>
    </errorInformation>
  </statusResults>
  <resourceURL>http://example.com/exampleAPI/messagebroadcast/v1/request/b0001/status</resourceURL>
</mb:status>
```

### 6.3.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: [GET]’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

### 6.3.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: [GET]’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

### 6.3.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: [GET]’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

# 7. Fault definitions

## 7.1 Service Exceptions

For common Service Exceptions refer to [REST_NetAPI_Common]. There are no additional Service Exception codes defined for the RESTful Message Broadcast API.

## 7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common]. There are no additional Policy Exception codes defined for the RESTful Message Broadcast API.

# Appendix A. Change History (Informative)

## A.1 Approved Version History

| Reference | Date | Description |
|---|---|---|
| n/a | n/a | No prior version |

## A.2 Draft/Candidate Version 1.0 History

| Document Identifier | Date | Sections | Description |
|---|---|---|---|
| Draft Versions:<br>REST_NetAPI _MsgBCast-V1_0 | 18 Jan 2016 | All | First draft |
| | 19 Feb 2016 | 1,<br>4 | Incorporated CR:<br>• OMA-REST_MsgBCast-2016-0002R01-CR_Content_for_Scope,_introduction_and_Version_1.0_sections_of_REST_MsgBCast_spec |
| | 29 Apr 2016 | 2.1,<br>5.1,<br>5.2,<br>5.3,<br>6 | Incorporated CRs:<br>• OMA-REST_MsgBCast-2016-0003R01-CR_Message_Broadcast_REST_API_resource_structure<br>• OMA-REST_MsgBCast-2016-0004R01-CR_Message_Broadcast_REST_API_resource_tree<br>• OMA-REST_MsgBCast-2016-0005R02-CR_Data_Types_for_message_broadcast<br>• OMA-REST_MsgBCast-2016-0006-CR_Sequence_diagrams_for_message_broadcast<br>• OMA-REST_MsgBCast-2016-0008R01-CR_XML_examples_for_request_resource_section_6 |
| | 10 May 2016 | 7,<br>Appendix B,<br>Appendix D<br>Appendix E<br>Appendix F | • OMA-REST_MsgBCast-2016-0007-CR_fault_definitions_for_section_7<br>• OMA-REST_MsgBCast-2016-0010R01-CR_SCR_tables<br>• OMA-REST_MsgBCast-2016-0013-CR_description_for_appendix_D_and_E<br>• OMA-REST_MsgBCast-2016-0014R01-CR_Content_for_appendix_F_authorisation_aspects |
| | 15 May 2016 | Front page | • Dates changed and aligned |

| Document Identifier | Date | Sections | Description |
|---|---|---|---|
| | 05 Jul 2016 | All | • OMA-REST_MsgBCast-2016-0015R01-CR_Resolution_of_editorial_review_comments_of_MsgBCast_API<br>• OMA-REST_MsgBCast-2016-0016R01-CR_Section_4_Introduction_MsgBCast_Concepts<br>• OMA-REST_MsgBCast-2016-0017R01-CR_Resolution_of_comments_A015_A018_of_MsgBCast_CONR<br>• OMA-REST_MsgBCast-2016-0018-CR_To_address_comment_A008_on_MsgBCast_1.0_operations<br>• OMA-REST_MsgBCast-2016-0019-CR_o_address_comments_A010_A012_of_MsgBCast_1.0_CONR<br>• OMA-REST_MsgBCast-2016-0020-CR_To_address_comments_A019__A020_of_MsgBCast_CONR<br>• OMA-REST_MsgBCast-2016-0021R01-CR_To_address_comments_A024__A036_of_MsgBCast_CONR<br>• OMA-REST_MsgBCast-2016-0022-CR_To_address_comments_A042_A044_of_MsgBCast_CONR<br>• OMA-REST_MsgBCast-2016-0023-CR_To_address_comment_A045_on_JSON_examples_for_MsgBCast<br>• Gf<br>• OMA-REST_MsgBCast-2016-0025-CR_To_address_comment_C001_on_ERELD_normative_references |
| | 13 Jul 2016 | Appendix D<br><br>Multiple sections | • OMA-REST_MsgBCast-2016-0024-CR_To_address_comment_A046_on_Parlay_X_operations_mapping<br>• OMA-REST_MsgBCast-2016-0027R01-CR_To_address_CONR_review_comment_A039 |
| | 27 Aug 2016 | All | • OMA-REST_MsgBCast-2016-0037-CR_Resolution_of_editorial_and_bug_fixes_of_ATT_Comments<br>• OMA-REST_MsgBCast-2016-0038-CR_Resolution_of_AT_T_comments_section_5.3<br>• OMA-REST_MsgBCast-2016-0039R01-CR_Resolution_of_AT_T_comments_for_section_6 |
| Candidate Version:<br>REST_NetAPI _MsgBCast-V1_0 | 13 Sep 2016 | n/a | Status changed to Candidate by TP<br>TP Ref # OMA-TP-2016-0095-INP_REST_NetAPI_MsgBCast_V1_0_ERP_for_Candidate_approval |

# Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

## B.1 SCR for REST.MsgBCast Server

| Item | Function | Reference | Requirement |
|---|---|---|---|
| REST-MsgBCast-SUPPORT-S-001-M | Support for the RESTful MsgBCast API | 5, 6 | |
| REST-MsgBCast-SUPPORT-S-002-M | Support for the XML request & response format | 6 | |
| REST-MsgBCast-SUPPORT-S-003-M | Support for the JSON request & response format | 6, Appendix C | |

### B.1.1 SCR for REST.MsgBCast.request Server

| Item | Function | Reference | Requirement |
|---|---|---|---|
| REST-MsgBCast-REQUEST-S-001-M | Support for allocation of resource for a message broadcast request and return to the client the newly created URL and support for retrieval of submitted requests list | 6.1 | |
| REST-MsgBCast-REQUEST-S-002-M | Support for querying about a list of submitted requests of message broadcast – GET | 6.1.3 | |
| REST-MsgBCast-REQUEST-S-003-M | Support for creation of a new request of message broadcast – POST | 6.1.5 | |

### B.1.2 SCR for REST.MsgBCast.requestId Server

| Item | Function | Reference | Requirement |
|---|---|---|---|
| REST-MsgBCast-REQUESTID-S-001-M | Support for retrieving, updating and deleting an individual message broadcast request | 6.2 | |
| REST-MsgBCast-REQUESTID-S-002-M | Retrieve information about a message broadcast request – GET | 6.2.3 | |
| REST-MsgBCast-REQUESTID-S-003-M | Update a message broadcast request – PUT | 6.2.4 | |
| REST-MsgBCast-REQUESTID-S-004-M | Delete a message broadcast request – DELETE | 6.2.6 | |

### B.1.3 SCR for REST.MsgBCast.status Server

| Item | Function | Reference | Requirement |
|---|---|---|---|
| REST-MsgBCast-STATUS-S-001-M | Support for retrieving the status of a message broadcast request | 6.3.3 | |

| Item | Function | Reference | Requirement |
|---|---|---|---|
| REST-MSGBCAST-STATUS-S002-M | Retrieve the status of message broadcast request – GET | 6.3.3 | |

# Appendix C. JSON examples (Informative)

JSON (JavaScript Object Notation) is a Light-weight, text-based, language-independent data interchange format. It provides a simple means to represent basic name-value pairs, arrays and objects. JSON is relatively trivial to parse and evaluate using standard JavaScript libraries, and hence is suited for REST invocations from browsers or other processors with JavaScript engines. Further information on JSON can be found at [RFC7159].

The following examples show the request and response for various operations using the JSON data format. The examples follow the XML to JSON serialization rules in [REST_NetAPI_Common]. A JSON response can be obtained by using the content type negotiation mechanism specified in [REST_NetAPI_Common].

For full details on the operations themselves please refer to the section number indicated.

## C.1 Retrieving a list of broadcast message requests (section 6.1.3.1)

Request:

```
GET /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "mb:requestList": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
      "request": [
      {
        "serial": "A00001EF",
        "broadcastArea": [
          {
            "unionElement": "Circle",
            "circle": {
              "centre": {
                "latitude": "51.5573",
                "longitude": "-0.3930"
              },
              "radius": "3000"
            }
          },
          {
            "unionElement": "Circle",
            "circle": {
              "centre": {
                "latitude": "51.5758",
                "longitude": "-0.4212"
              },
              "radius": "2000"
            }
```

```
        }
       ],
       "senderName": "South Ruislip Traffic Police",
       "charging": {
        "description": "Test amount transaction \"Charged\"",
        "currency": "GBP",
        "amount": "10",
        "code": "TEST-012345"
       },
       "message": "Major Traffic Accident at the Polish War Memorial",
       "priority": "High",
       "deliveryTime": "2016-06-23T18:45:00-07:00",
       "totalBroadcasts": "15",
       "interval": "7200",
       "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
      },
      {
       "serial": "86066117",
       "broadcastArea": [
        {
        "unionElement": "Circle",
        "circle": {
         "centre": {
          "latitude": "51.6054",
          "longitude": "-0.1222"
         },
         "radius": "2000"
        }
       },
       ],
       "senderName": "Harringay Fire Brigade",
       "message": "Building on fire",
       "priority": "High",
       "deliveryTime": "2016-03-26T18:45:00-07:00",
       "totalBroadcasts": "10",
       "interval": "7200",
       "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0002"
      },
    ],
    "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request"
  }
}
```


## C.2 Retrieving a list of broadcast message requests, response with a list of resource references (section 6.1.3.2)

Request:


```
GET /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/json
```


Response:

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/json
Content-Length: nnnn
{
  "mb:requestList": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "requestReference": [
      { "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001" },
      { "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0002" }
    ]
  }
}
```

## C.3 Requesting to send a broadcast message in specified geographic areas (section 6.1.5.1)

Request:

```
POST /exampleAPI/messagebroadcast/v1/request HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: nnnn
MIME-Version: 1.0

{
  "mb:request": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "serial": "A00001EF",
    "broadcastArea": [
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5573",
            "longitude": "-0.3930"
          },
          "radius": "3000"
        }
      },
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      }
    ],
    "senderName": "South Ruislip Traffic Police",
    "charging": {
      "description": "Test amount transaction \"Charged\"",
```

```
    "currency": "USD",
    "amount": "10",
    "code": "TEST-012345"
   },
   "message": "Major Traffic Accident at the Polish War Memorial",
   "priority": "High",
   "deliveryTime": "2016-06-23T18:45:00-07:00",
   "totalBroadcasts": "15",
   "interval": "7200"
  }
}
```

Response:

```
HTTP/1.1 201 Created
Date: Mon, 07 Mar 2016 15:17:02 GMT
Location: http://example.com/exampleAPI/messagebroadcast/v1/request/b0001
Content-Type: application/json
Content-Length: nnnn

{
  "mb:request": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "serial": "A00001EF",
    "broadcastArea": [
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5573",
            "longitude": "-0.3930"
          },
          "radius": "3000"
        }
      },
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      }
    ],
    "senderName": "South Ruislip Traffic Police",
    "charging": {
      "description": "Test amount transaction \"Charged\"",
      "currency": "USD",
      "amount": "10",
      "code": "TEST-012345"
    },
    "message": "Major Traffic Accident at the Polish War Memorial",
    "priority": "High",
    "deliveryTime": "2016-06-23T18:45:00-07:00",
```

```
   "totalBroadcasts": "15",
   "interval": "7200"
   "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
  }
}
```

# C.4 Retrieve a submitted message broadcast request (section 6.2.3.1)

Request:

```
GET /exampleAPI/messagebroadcast/v1/request/b0001 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 7 Mar 2016 10:50:00 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "mb:request": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "serial": "A00001EF",
    "broadcastArea": [
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5573",
            "longitude": "-0.3930"
          },
          "radius": "3000"
        }
      },
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      }
    ],
    "senderName": "South Ruislip Traffic Police",
    "charging": {
      "description": "Test amount transaction \"Charged\"",
      "currency": "USD",
      "amount": "10",
```

```
    "code": "TEST-012345"
   },
   "message": "Major Traffic Accident at the Polish War Memorial",
   "priority": "High",
   "deliveryTime": "2016-06-23T18:45:00-07:00",
   "totalBroadcasts": "15",
   "interval": "7200",
   "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
  }
}
```

## C.5 Update submitted message broadcast request (section 6.2.4.1)

Request:

```
PUT /exampleAPI/ messagebroadcast/v1/request/b0001 HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: nnnn
MIME-Version: 1.0

{
  "mb:request": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "serial": "A00001EF",
    "broadcastArea": [
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5573",
            "longitude": "-0.3930"
          },
          "radius": "4000"
        }
      },
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      }
    ],
    "senderName": "South Ruislip Traffic Police",
    "charging": {
      "description": "Test amount transaction \"Charged\"",
      "currency": "USD",
      "amount": "10",
      "code": "TEST-012345"
```

```
    },
    "message": "Major Traffic Accident at the Polish War Memorial",
    "priority": "High",
    "deliveryTime": "2016-06-23T18:45:00-07:00",
    "totalBroadcasts": "15",
    "interval": "7200",
    "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
  }
}
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 07 Mar 2016 11:00:00 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "mb:request": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "serial": "A00001EF",
    "broadcastArea": [
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5573",
            "longitude": "-0.3930"
          },
          "radius": "4000"
        }
      },
      {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      }
    ],
    "senderName": "South Ruislip Traffic Police",
    "charging": {
      "description": "Test amount transaction \"Charged\"",
      "currency": "USD",
      "amount": "10",
      "code": "TEST-012345"
    },
    "message": "Major Traffic Accident at the Polish War Memorial",
    "priority": "High",
    "deliveryTime": "2016-06-23T18:45:00-07:00",
    "totalBroadcasts": "15",
    "interval": "7200",
    "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
```

```
 }
}
```

## C.6 Delete submitted message broadcast request (section 6.2.6.1)

Request:

```
DELETE/exampleAPI/messagebroadcast/v1/request/b0001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 204 No Content
Date: Date: Tue, 7 Mar 2016 10:50:00 GMT
```

## C.7 Retrieve the message broadcast request status (section 6.3.3.1)

Request:

```
GET /exampleAPI/messagebroadcast/v1/request/b0001/status HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 7 Mar 2016 10:50:00 GMT
Content-Type: application/json
Content-Length: nnnn


{
  "mb:status": {
    "-xmlns:mb": "urn:oma:xml:rest:netapi:messagebroadcast:1",
    "link": {
      "-rel": "requestReference",
      "-href": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001"
    },
    "statusResults": [
      {
        "area": {
          "unionElement": "Circle",
          "circle": {
            "centre": {
              "latitude": "51.6054",
              "longitude": "-0.1222"
            },
            "radius": "2000"
          }
        },
```

```
      "reportStatus": "Retrieved",
      "currentStatus": {
        "status": "Broadcasted",
        "numberOfBroadcasts": "15",
        "successRate": "100",
        "broadcastEndTime": "2016-03-26T18:45:00-00:00"
      }
    },
    {
      "area": {
        "unionElement": "Circle",
        "circle": {
          "centre": {
            "latitude": "51.5758",
            "longitude": "-0.4212"
          },
          "radius": "2000"
        }
      },
      "reportStatus": "Retrieved",
      "currentStatus": {
        "status": "BroadcastImpossible",
        "numberOfBroadcasts": "0",
        "successRate": "0",
        "broadcastEndTime": "2016-03-26T21:32:00-00:00"
      },
      "errorInformation": {
        "messageId": "SVC0300",
        "text": "Broadcast Area not supported"
      }
    }
  ],
  "resourceURL": "http://example.com/exampleAPI/messagebroadcast/v1/request/b0001/status"
 }
}
```

# Appendix D. Parlay X Operations (Informative)

The table below illustrates the mapping between REST resources/methods defined in this specification and Parlay X [3GPP 29.199-15] equivalent operations.

| **REST Resource** | **REST Method** | **REST Section reference** | Parlay X **equivalent operation** |
|---|---|---|---|
| Message broadcast request | GET | 6.1.3 | N/A |
| Message broadcast request | POST | 6.1.5 | SendBroadcastMessage |
| Message Broadcast Request Info | GET | 6.2.3 | N/A |
| Message Broadcast Request Info | PUT | 6.2.4 | SendBroadcastMessage |
| Message Broadcast Request Info | DELETE | 6.2.6 | cancelBroadcastMessageRequest |
| Message broadcast request status | GET | 6.3.3 | getBroadcastStatus |

**Table 1: Parlay X operations mapping**

# Appendix E. Light-weight Resources (Informative)

As this version of the specification does not define any Light-weight Resources, this appendix is empty.

# Appendix F. Authorization aspects (Normative)

This appendix specifies how to use the RESTful Message BroadcastAPI in combination with some authorization frameworks.

## F.1 Use with OMA Authorization Framework for Network APIs

The RESTful Message Broadcast API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Message Broadcast

API supporting [Autho4API_10]:

- SHALL conform to section D.1 of [REST_NetAPI_Common];
- SHALL conform to this section G.1.

### F.1.1 Scope values

#### F.1.1.1 Definitions

In compliance with [Autho4API_10], an authorization server serving clients requests for getting authorized access to the resources exposed by the RESTful Message Broadcast API:

- SHALL support the scope values defined in the table below;
- MAY support scope values not defined in this specification.

| Scope value | Description | For one-time access token |
|---|---|---|
| oma_rest_messagebroadcast.all_{apiVersion} | Provide access to all defined operations on the resources in this version of the API. The {apiVersion} part of this identifier SHALL have the same value as the “apiVersion” URL variable which is defined in section 5.1. This scope value is the union of the other scope values listed in next rows of this table. | No |
| oma_rest_messagebroadcast_request | Provides access to all defined operations for message broadcast request(s) except status | No |
| oma_rest_messagebroadcast.status | Provides access to defined operations on status resource | No |

**Table 2: Scope values for RESTful Message BroadcastAPI**

#### F.1.1.2 Downscoping

In the case where the client requests authorization for “oma_rest_messagebroadcast.all_{apiVersion}” scope, the authorization server and/or resource owner MAY restrict the granted scope to some of the following scope values:

- oma_rest_messagebroadcast.request

#### F.1.1.3 Mapping with resources and methods

| Resource | URL Base URL: http://{serverRoot}/MessageBroadcast/{apiVersion} | Section reference | HTTP verbs | | | |
|---|---|---|---|---|---|---|
| | | | GET | PUT | POST | DELETE |
| Message Broadcast Requests | /request | 6.1 | all_{apiVersion} or request | no | all_{apiVersion} or request | no |
| Individual message broadcast request | /request/{requestId} | 6.2 | all_{apiVersion} or request | all_{apiVersion} or request | no | all_{apiVersion} or request |
| Message broadcast request status | /request/{requestId}/status | 6.3 | all_{apiVersion} or status | no | no | no |

**Table 3: Required scope values for: Message Broadcast request management and request status enquiry**

## F.1.2 Use of 'acr:auth'

This section specifies the use of 'acr:auth' in place of an end user identifier in a resource URL path.

An 'acr' URI of the form 'acr:auth', where 'auth' is a reserved keyword MAY be used to avoid exposing a real end user identifier in the resource URL path.

Note: 'acr:auth' in place of the end user identifier part of a resource URL path is not used in this specification since end user identifier (e.g. {userId}, {address}, etc.) is not part of the resource URL path defined in this specification.