RESTful Network APIs for Bot Management

Candidate Version 1.0 – 29 May 2018

Open Mobile Alliance

OMA-TS-REST_NetAPI_BotManagement-V1_0-20180529-C

Contents

Figures

Tables

1. Scope

This specification defines a RESTful API for Bot Management using HTTP protocol bindings.

2. References

2.1 Normative References

[Autho4API_10]	"Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL: http://www.openmobilealliance.org/
[RCC_API_RD]	GSMA RCC.13 RCS API Detailed Requirements v3.0
[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_SUP_BotManagemet]	"XML schema for the RESTful Network API for Bot Management", Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_botmanagement-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
[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
[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
[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].

Chatbot	An RSC-based service provided to users whose output is presented in a conversational form and which provides some kind of value to the users. Often a piece of software interfacing with one or more users aiming to simulate intelligent human conversation
Chatbot Platform	A system that provides a mechanism for Chatbot developers to create and register bots, which can then be exposed to the users connected to the platform through a messaging system.

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 APIs for Bot Management contains HTTP protocol bindings for Bot Management, 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).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations:

- Deleting a user’s Anonymization token
- Managing subscriptions to Spam report notifications

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. Bot Management API definition

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

This specification enables two separate APIs both of which are related to bot management.

The Bot Management API enables the following:

A- Enable an authorized client (i.e. Service Provider's API Gateway) to delete an anonymization token when anonymization function is deployed by the server (i.e. Chatbot Platform)

B- Enable an authorized client (e.g. Chatbot Platform) to subscribe/unsubscribe to chatbot spam report notifications exposed by the server (Service Provider's notification server) when spam report function is deployed within Service Provider Network
.

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 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 APIs for Bot Management.

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.

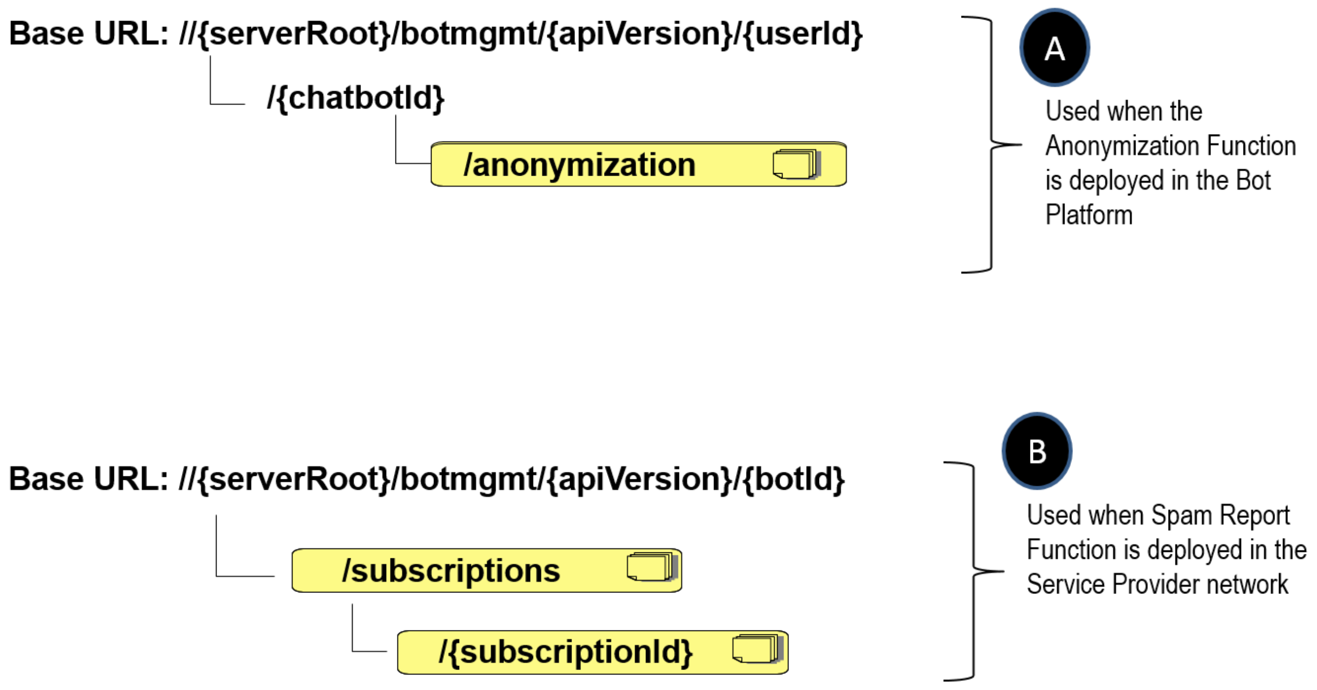
Base URL: //{serverRoot}/botmgmt/{apiVersion}/{userId}
/{chatbotId}
/anonymization
A
Used when the Anonymization Function is deployed in the Bot Platform
Base URL: //{serverRoot}/botmgmt/{apiVersion}/{botId}
/subscriptions
/{subscriptionId}
B
Used when Spam Report Function is deployed in the Service Provider network

Figure 1: Resource structure defined by this specification

This specification defines resource structure in support of both Anonymization Function as well as Spam Report Function. Anonymization related resources is used when the Anonymization Function is deployed in the Chatbot Platform (i.e. see “A” in above figure). That is the Service Provider’s API Gateway plays the role of the client of the API.

On the other hand, the Spam Report related resources is used when Spam Report Function is deployed in the Service Provider network (i.e. “B” resources). That is the Chatbot Platform plays the role of the client of the API

Note: “**subscriptions**” is a reserved keyword and MUST NOT be used as a {chatbotId}.

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 a client to manage (delete) anonymization Function

Resource	URL Base URL: //{serverRoot}/botmgmt/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
User's anonymization	/{userId}/{chatbotId}/anonymization	Empty	no	no	Delete anonymization token	no

Purpose: To allow client to manage subscriptions to notifications (e.g. spam report) related to chatbots

Resource	URL Base URL: //{serverRoot}/botsmgmt/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
All subscriptions to bot related notifications	/{botId}/subscriptions	BotSubscriptionList (used for GET) BotSubscription (used for POST) common:ResourceReference (OPTIONAL alternative for POST response)	Retrieve all active bot related notification subscriptions	no	Create new subscription for notifications related to bot(s)	no
Individual subscription	/{botId}/subscriptions/{subscriptionId}	BotSubscription (used for GET response)	Retrieve an individual subscription	no	no	Cancel subscription and stop corresponding notifications

Purpose: To allow server to notify client about spam reports

Resource	URL<specified by the client>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE

Resource	URL<specified by the client>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Client notification about spam reports	Specified by client when subscription is created or provisioned	SpamReportNotification	no	no	Notifies client about the spam reports	no

5.2 Data Types

5.2.1 XML Namespaces

The XML namespace for the Bot Management data types is:

urn:oma:xml:rest:netapi:botmanagement: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_BotManagement].

5.2.2 Structures

The subsections of this section define the data structures used in the Bot Management 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: BotSubscriptionList

This type represents a list of subscriptions to notifications regarding bot related events.

Element	Type	Optional	Description
subscription	BotSubscription[0..unbounded]	Yes	Array of notification subscriptions.
resourceURL	xsd:anyURI	No	Self referring URL.

A root element named botSubscriptionList of type BotSubscriptionList is allowed in response bodies.

5.2.2.2 Type: BotSubscription

This type represents a subscription to notifications regarding bot events (e.g. spam report event).

Element	Type	Optional	Description
callbackReference	common:CallbackReference	No	Client's Notification URL and OPTIONAL callbackData.
duration	xsd:unsignedInt	Yes	Period of time (in seconds) notifications are provided for. If set to “0” (zero), a default duration time, which is specified by the service policy, will be used. If the parameter is omitted, the notifications will continue until the maximum duration time, which is specified by the service policy, unless the notifications are stopped by deletion of subscription for notifications. This element MAY be given by the client during resource creation in order to signal the desired lifetime of the subscription. The server SHOULD return in this element the period of time for which the subscription will still be valid.

listId	xsd:anyURI	Yes	The element is pointing to a list available on the API Server containing identifiers (e.g. bot applications identifiers). This element MUST be present in POST if the request URL variable {botId} is set to an aggregator identifier (e.g. chatbot platform Id). For example if a chatbot platform desires to create a single subscription for a list of chabots as opposed to creating one subscription per chatbot, the listId must be provided in the subscription creation request in order to identify the chatbots which are served by the given subscription (see section 6.2.5.1 and 6.2.5.2)
clientCorrelator	xsd:string	Yes	A correlator that the client can use to tag this particular resource representation during a request to create a resource on the server. This element MAY be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate subscriptions in such situations. In case the element is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
resourceURL	xsd:anyURI	Yes	Self referring URL. 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 botSubscription of type BotSubscription is allowed in request and/or response bodies

5.2.2.3 Type: Empty

A dummy element

Element	Type	Optional	Description
(empty)		No	This type is empty and used as a dummy element, to overcome some limitations, for example: 1. HTTP POST requires a body, but the semantics of the request/response may not need that body. 2. Some implementations may not be able to support non-existent (empty) HTTP body.

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

5.2.2.4 Type: SpamReportInfo

This type represents information a spam report

Element	Type	Optional	Description
userId	xsd:anyURI	No	The address (e.g. 'sip' URI,'tel' URI, 'acr' URI) of the user.
chatbotId	xsd:anyURI	No	The address (e.g. 'sip' URI,'tel' URI, 'acr' URI) of the chatbot.
messageId	xsd:string[0..10]	Yes	List of message identifiers being reported (by the userId) as spam (received from the chatbotId)

5.2.2.5 Type: SpamReportNotification

This type represents a notification about a spam report.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The 'callbackData' element if it was passed by the application in the 'callbackReference' element when creating a subscription to notifications. See [REST_NetAPI_Common]
spamReportInfo	SpamReportInfo	No	Spam report information identifying the user reporting the spam messages received from a chatbot.
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship to the notification. Further, the server SHOULD include a link to the related subscription.

A root element named spamReportNotification of type SpamReportNotification is allowed in notification

5.2.3 Enumerations

There is no defined enumerations in the Bot Management API.

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):

- BotSubscription

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 Subscription to notifications

This figure below shows a scenario for an API client (e.g. Chatbot Plaform) subscribing to notifications (e.g. spam report), querying for a list of active subscriptions, querying information pertaining to a subscription and unsubscribing to notifications. While the Spam Report Function can be deployed in different entities in the network (e.g. Service Provider API Gateway or Chatbot Platform), in this flow, it is assumed that the Service Provider API Gateway owning the Spam Report Function.

The notification URL passed by the client during the subscription step is a Client-side Notification URL.

The resources:

- To subscribe to bot related notifications, create a new resource under **http://{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions**
- To retrieve list of active subscriptions, read the following resource **http://{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions**
- To retrieve information about an individual subscription, read the following resource **http://{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions/{subscriptionId}**
- To cancel subscription to bot related notifications delete the resource under **http://{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions/{subscriptionId}**

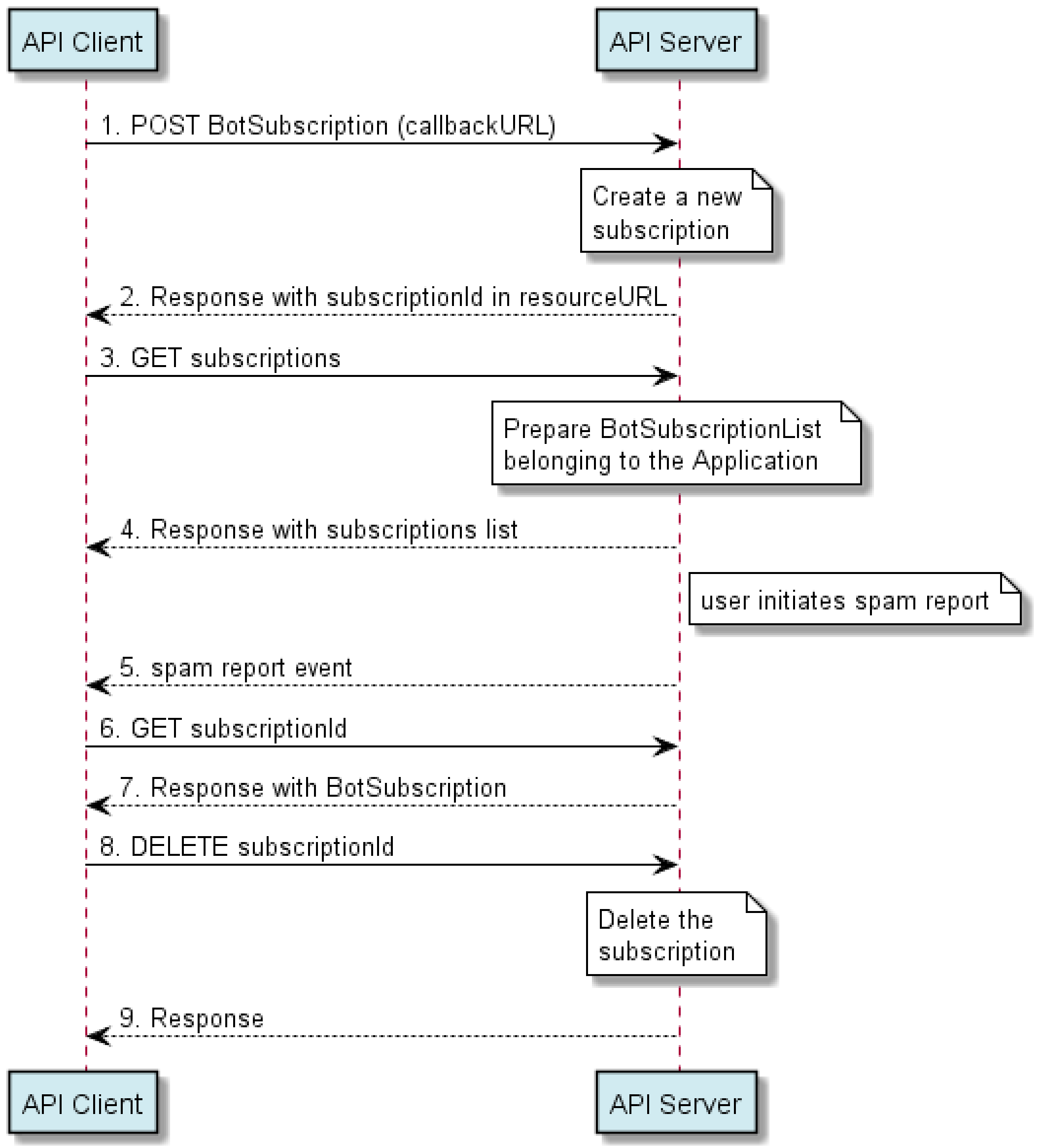
API Client
API Server
1. POST BotSubscription (callbackURL)
Create a new subscription
2. Response with subscriptionId in resourceURL
3. GET subscriptions
Prepare BotSubscriptionList belonging to the Application
4. Response with subscriptions list
user initiates spam report
5. spam report event
6. GET subscriptionId
7. Response with BotSubscription
8. DELETE subscriptionId
Delete the subscription
9. Response
API Client
API Server

Figure 2 Notification subscription

Outline of the flows:

1. The API client (e.g. Chatbot Platform) subscribes to bot related notifications (e.g. spam report event) using the POST method to submit the BotSubscription data structure to the resource containing all subscriptions.
2. The API client receives the result resource URL containing the subscriptionId.
3. The API client requests the list of active subscriptions, using the GET method to the resource containing all subscriptions.
4. The API server returns subscriptions list belonging to the application in the response.
5. User initiates an spam report (out of scope of this API) and as a result, the API server (e.g. Service Provider API Gateway) which implements Spam Report Functionality, reports this event to the application's notification URL which was passed in step 1.
6. The API client requests information pertaining to an individual subscription using the GET method to the resource.
7. The API server returns subscription's data which includes duration, etc. in the response.

8. The application stops receiving notifications (on a given subscription) using DELETE with the resource URL containing the subscriptionId.
9. Deletion confirmation.

5.3.2 Anonymization Management Operations

This figure below shows a scenario for an API client application on behalf of the user (e.g. RCS client) delete the anonymization token which the server is using to mask user’s true identity (e.g. MSISDN) in its communication with an online service (e.g. a chatbot).

The resources:

- To delete the anonymization token used by the server to hide the user’s true identity in its communication with a 3rd party online service (e.g. a chatbot) action the following resource **http://{serverRoot}/botmgmt/{apiVersion}/{userId}/{chatbotId}/anonymization**

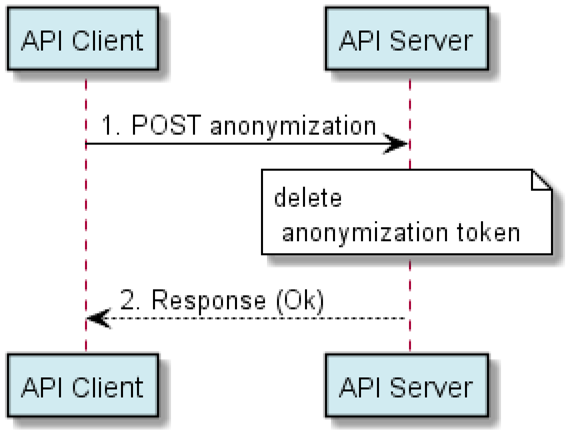

Figure 3 Anonymization management operation

Outline of the flows:

Based on user’s request

1. The API Client requests anonymization deletion operation (through POST to anonymization resource)
2. The API server removes the anonymization token on the server and acknowledges the operation result back to the API client.

Note[1]: How an anonymization token was created on the server side (e.g. Chatbot Platform) to begin with is outside of the scope of this API.

[1] An anonymization token (i.e. a token resource instance) is created through an out-of-band (non-RESTful API) method. As a result, using DELETE operation is not appropriate as no assumption can be made on the existence of a RESTful anonymization token resource (i.e. token instanceId).

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: User's anonymization

The resource used is:

http://{serverRoot}/botmgmt/{apiVersion}/{userId}/{chatbotId}/anonymization

This resource is used by a client on behalf of the user to delete the anonymization token used by the server in its attempt to hide the user's true identity in its communication with a 3rd party online service (e.g. a chatbot).

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. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
chatbotId	Identifier of the chatbot which the user is in communication with Examples: tel:+19585550101, sip:onlineChatbot1@example.com

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 Bot Management, see section 7.

6.1.3 GET

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

6.1.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: [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 to delete the anonymization token employed by the server in its attempt to hide the user's true identity in its communication with a 3rd party online service (e.g. a chatbot). The anonymization token is created through an out-of-band (i.e. non-RESTful API) method which is outside the scope of this specification.

6.1.5.1 Example 1: Client application (e.g. API Gateway) on behalf of user deletes anonymization token on the server (e.g. Chatbot Platform) (Informative)

6.1.5.1.1 Request

```
POST /exampleAPI/botmgmt/v1/tel%3A%2B19585550100/sip%3bot3%40example.com/anonymization HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:empty xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1"/>
```

6.1.5.1.2 Response

```
HTTP/1.1 200 Ok
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:empty xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1"/>
```

6.1.6 DELETE

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

6.2 Resource: All subscriptions to bot related notifications

The resource used is:

//{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions

This resource is used to manage subscriptions to event notifications related to Bot Management (e.g. spam report event).

This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

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
botId	Identifier of a service (e.g. chatbot) or a service aggregator (e.g. Chatbot Platform) For Example: sip:bot42@example.com, sip:aggr101@example.com, sip:botplatform5@example.com)

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

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 RESTful Network APIs for Bot Management, see section 7.

6.2.3 GET

This operation is used for reading the list of active notification subscriptions. Note that only the subscriptions this client is authorised to access SHALL be included in the list.

6.2.3.1 Example: Reading all active subscriptions (Informative)

Application client (e.g. Chatbot Platform) reads all active subscriptions.

6.2.3.1.1 Request

```
GET /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: Bearer 08776724-6d0d-4aa6-a404-2bc19b5cf903
```

6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:botSubscriptionList xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
    <subscription>
        <callbackReference>
            <notifyURL>http://applicationClient.example.com/spamReport/notifications/77777</notifyURL>
            <callbackData>abcd</callbackData>
        </callbackReference>
 <listId>/botplat1/chatbotList3</listId>
 <clientCorrelator>12345</clientCorrelator>
        <resourceURL>http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions
/sub001</resourceURL>
    </subscription>
    <resourceURL>http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions</resourceURL>
```

```
</botmgmt:botSubscriptionList>
```

6.2.4 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 section 6.5.5 and 7.4.1 of [RFC7231].

6.2.5 POST

This operation is used to create a new subscription for notifications related to Bot events (e.g. spam reports).

The notifyURL in the callbackReference either contains the Client-side Notification URL (as defined by the client) or the Server-side Notification URL (as obtained during the creation of the Notification Channel [REST_NetAPI_NotificationChannel]).

6.2.5.1 Example 1: Creating a new subscription by a Service Aggregator client, response with copy of created resource (Informative)

The following example shows a subscription creation request containing a “listId”. This indicates that the {botId} in the request URL is the Id of a service aggregator (e.g. chatbot Platform) which wishes to create this one subscription to handle all the events related to many services (such as chatbots which are identified by serviceIds listed in the “listId) it is aggregating.

6.2.5.1.1 Request

```
POST /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:botSubscription xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/spamReport/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
   <listId>sip:chatbot_list_for_CBPx@example.com</listId>
   <clientCorrelator>12345</clientCorrelator>
</botmgmt:botSubscription>
```

6.2.5.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT
<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:botSubscription xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/spamReport/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
 <listId>sip:chatbot_list_for_CBPx@example.com</listId>
<clientCorrelator>12345</clientCorrelator>
<resourceURL>http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001</resourceURL>
</botmgmt:botSubscription>
```

6.2.5.2 Example 2: Creating a new subscription, response with location of created resource (Informative)

Application client creates a subscription. Note, since the {botId} of the request URL is the Id of the actual service and not the aggregator itself, “listId” element is not required in the POST request body.

6.2.5.2.1 Request

```
POST /exampleAPI/botmgmt/v1/sip%3Abot9%40example.com/subscriptions HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:botSubscription xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/spamReport/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
<clientCorrelator>12345</clientCorrelator>
</botmgmt:botSubscription>
```

6.2.5.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/botmgmt/v1/sip%3Abot9%40example.com/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
   <resourceURL>http://example.com/exampleAPI/botmgmt/v1/sip%3Abot9%40example.com/subscriptions/sub001</resourceURL>
</common:resourceReference>
```

6.2.6 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 section 6.5.5 and 7.4.1 of [RFC7231].

6.3 Resource: Individual subscription

The resource used is:

//{serverRoot}/botmgmt/{apiVersion}/{botId}/subscriptions/{subscriptionId}

This resource is used to manage an individual event subscription. This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

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. Example: example.com/exampleAPI

apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1
botId	Identifier of a service (e.g. chatbot) or a service aggregator (e.g. Chatbot Platform) For Example: sip:bot42@example.com, sip:aggr101@example.com, sip:botplatform5@example.com)
subscriptionId	Identifier of the subscription

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 the RESTful Network APIs for Bot Management, see section 7.

6.3.3 GET

This operation is used for reading an individual subscription.

6.3.3.1 Example 1: Reading an individual subscription (Informative)

Application client reads a subscription.

6.3.3.1.1 Request

```
GET /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: Bearer 08776724-6d0d-4aa6-a404-2bc19b5cf903
```

6.3.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<botmgmt:botSubscription xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/spamReport/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
 <listId>sip:chatbot_list_for_CBPx@example.com</listId>
<clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001</resourceURL>
</botmgmt:botSubscription>
```

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, DELETE’ field in the response as per section 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, DELETE’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.3.6 DELETE

This operation is used to cancel a subscription and to stop corresponding notifications.

6.3.6.1 Example: Cancelling a subscription (Informative)

Application client cancels a subscription.

6.3.6.1.1 Request

```
DELETE /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.3.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Tue, 17 Oct 2017 21:32:52 GMT
```

6.4 Resource: Client notification about Spams

This resource is a callback URL provided by the client for notification about spams (a user is receiving from a given bot). The RESTful Bot Management API does not make any assumption about the structure of this URL. The assumption is that this URL is a Client-side Notification URL and hence, the server will POST notifications directly to it.

6.4.1 Request URL variables

Client provided if any.

6.4.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Bot Management see section 7.

6.4.3 GET

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

6.4.4 PUT

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

6.4.5 POST

This operation is used for spam report notifications (i.e. notifying the client about spams a given user is receiving from a given bot).

6.4.5.1 Example 1: Notify client about spams the user is receiving from a bot (Informative)

6.4.5.1.1 Request

```
POST /spamReport/notifications/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: applicationClient.example.com

<?xml version="1.0" encoding="UTF-8"?>
```

```
<botmgmt:spamReportNotification xmlns:botmgmt="urn:oma:xml:rest:netapi:botmanagement:1">
   <callbackData>abcd</callbackData>
    <spamReportInfo >
        <userId>tel:+19585550101</userId>
        <chatbotId>sip:bot42@example.com</chatbotId>
        <messageId>msg10</messageId>
        <messageId>msg8</messageId>
   </spamReportInfo>
  <link rel="BotSubscription"
         href="http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001"/>
</botmgmt:spamReportNotification >
```

6.4.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Fri, 20 Oct 2017 17:51:59 GMT
```

6.4.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ 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 Bot Management 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: REST_NetAPI_AliasManagement-V1_0	22 Sep 2017	All	First draft
	10 Oct 2017	1, 2.1, 2.2, 3.2, 4, 4.1, 5, 5.1	OMA-ARC-Alias-2017-0006-CR_TS_Intro_sections OMA-ARC-Alias-2017-0007R01-CR_TS_sec5_resource_summary OMA-ARC-Alias-2017-0008-CR_TS_sec5_API_def OMA-ARC-Alias-2017-0010-CR_Refs_Defs
	12 Oct 2017	2.1, 5.1, 5.2	OMA-ARC-Alias-2017-0011R03-CR_resource_tree_subscription_improvement OMA-ARC-Alias-2017-0012R01-CR_data_types
	07 Nov 2017	5.0, 5.1, 5.2, 5.3, 5.3.1, 5.3.2, 5.2.2.4, 5.1, 5.2.2.2, 5.2.2.5, 5.2.2.7, 5.2.2.8, 6.1, 6.2, 6.3, 6.4, 6.5, Appendix B,	OMA-ARC-Alias-2017-0014R01-CR_Subscription_Sequence_Diagram, OMA-ARC-Alias-2017-0027R02-CR_Notification_step_added_subscription_flow, OMA-ARC-Alias-2017-0013-CR_small_cleanups, OMA-ARC-Alias-2017-0026R01-CR_apply_resource_tree_enhancements, OMA-ARC-Alias-2017-0024-CR_data_type_cleanup, OMA-ARC-Alias-2017-0023-CR_Spam_Report_notification, OMA-ARC-Alias-2017-0022-CR_Provide_Appendix_B_SCR, OMA-ARC-Alias-2017-0021-CR_XML_examples_SubscriptionId_resource, OMA-TS-REST_NetAPI_AliasManagement-V1_0-20171012-D_CR20, OMA-TS-REST_NetAPI_AliasManagement-V1_0-20171012-D_CR16R02, OMA-TS-REST_NetAPI_AliasManagement-V1_0-20171012-D_CR17, OMA-TS-REST_NetAPI_AliasManagement-V1_0-20171012-D_CR19 OMA-TS-REST_NetAPI_AliasManagement-V1_0-20171012-D_CR15R02
	16 Nov 2017	Almost all sections	OMA-ARC-Alias-2017-0028-CR_TS_nameChange_ResourceTree_clarification OMA-ARC-Alias-2017-0029R02-CR_botId_listId_XML_example_clarifications OMA-ARC-Alias-2017-0030-CR_SCR_edits OMA-ARC-Alias-2017-0031-CR_AppendixF_Authorization_aspect OMA-ARC-Alias-2017-0032-CR_Add_JSON_examples OMA-ARC-Alias-2017-0034-CR_XML_error_examples_for_aliasLink OMA-ARC-Alias-2017-0035-CR_new_Appendix_showing_E2E_flow OMA-ARC-Alias-2017-0036R01-CR_listId_description_clarification
	24 Apr 2018	Many sections based on E2E consistency review	OMA-ARC-Alias-2018-0004R01-CR_ERELD_Cleanups
Draft Versions: REST_NetAPI_BotManagement-V1_0	25 Apr 2018	n/a	Fixed the file name before Candidate approval
	22 May 2018	C.7	OMA-ARC-Bot-2018-0009-CR_Add_missing_JSON_example
Candidate Version: REST_NetAPI_BotManagement-V1_0	29 May 2018	All	Status changed to Candidate by ARC Doc Ref # OMA-ARC-2018-0018-INP_REST_NetAPI_Bot_Management_V1_0_ERP_for_Candidate_approval

Appendix B. Static Conformance Requirements (Normative)

No interoperability testing is foreseen within OMA.

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 Client application (e.g. API GW) on behalf of user deletes anonymization token on the server (e.g. Chatbot Platform) (section 6.1.5.1)

Request:

```
POST /exampleAPI/botmgmt/v1/tel%3A%2B19585550100/sip%3bot3%40example.com/anonymization HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"empty": null}
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

{"empty": null}
```

C.2 Reading all active subscriptions (section 6.3.3.1)

Request:

```
GET /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

{"botSubscriptionList": {
  "resourceURL": "http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions",
  "subscription": {
        "callbackReference": {
            "callbackData": "abcd",
            "notifyURL": "http://applicationClient.example.com/spamReport/notifications/77777"
        },
        "clientCorrelator": "12345",
        "listId": "sip:chatbot_list_for_CBPx@example.com",
        "resourceURL": "http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions /sub001"
    }
}}
```

C.3 Creating a new subscription by a Service Aggregator client, response with copy of created resource (section 6.2.5.1)

Request:

```
POST /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"botSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/spamReport/notifications/77777"
    },
    "clientCorrelator": "12345",
    "listId": "sip:chatbot_list_for_CBPx@example.com"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

{"botSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/spamReport/notifications/77777"
    },
    "clientCorrelator": "12345",
    "listId": "sip:chatbot_list_for_CBPx@example.com",
    "resourceURL": "http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001"
}}
```

C.4 Creating a new subscription, response with location of created resource (section 6.2.5.2)

Request:

```
POST /exampleAPI/botmgmt/v1/sip%3Abot9%40example.com/subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"botSubscription": {
   "callbackReference": {
      "callbackData": "abcd",
      "notifyURL": "http://applicationClient.example.com/spamReport/notifications/77777"
   },
   "clientCorrelator": "12345"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/botmgmt/v1/sip%3Abot9%40example.com/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

{"resourceReference": {"resourceURL": "http://example.com/exampleAPI/botmgmt/v1/
sip%3Abot9%40example.com/subscriptions/sub001"}}
```

C.5 Reading an individual subscription (section 6.3.3.1)

Request:

```
GET /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 17 Oct 2017 21:32:52 GMT

{"botSubscription": {
   "callbackReference": {
      "callbackData": "abcd",
      "notifyURL": "http://applicationClient.example.com/spamReport/notifications/77777"
   },
   "clientCorrelator": "12345",
   "listId": "sip:chatbot_list_for_CBPx@example.com",
   "resourceURL": "http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001"
}}
```

C.6 Cancelling a subscription (section 6.3.6.1)

Request:

```
DELETE /exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001 HTTP/1.1
Content-Type: application/json
Host: example.com
```

Response:

```
HTTP/1.1 204 No Content
Date: Tue, 17 Oct 2017 21:32:52 GMT
```

C.7 Notify client about spams the user is receiving from a bot (section 6.4.5.1)

Request:

```
POST /spamReport/notifications/77777 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: applicationClient.example.com

{"spamReportNotification": {
    "callbackData": "abcd",
    "link": {
        "href": "http://example.com/exampleAPI/botmgmt/v1/sip%3Abotplat1%40example.com/subscriptions/sub001",
        "rel": "BotSubscription"
    },
    "spamReportInfo": {
        "chatbotId": "sip:bot42@example.com",
        "messageId": [
            "msg10",
            "msg8"
        ],
        "userId": "tel:+19585550101"
    }
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Fri, 20 Oct 2017 17:51:59 GMT
```

Appendix D. Operations mapping to a pre-existing baseline specification (Informative)

As this specification does not have a baseline specification, this appendix is empty.

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 Bot Management API in combination with some authorization frameworks.

F.1 Use with OMA Authorization Framework for Network APIs

The RESTful Bot Management API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Bot Management API supporting [Autho4API_10]:

- SHALL conform to section D.1 of [REST_NetAPI_Common];
- SHALL conform to this section F.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 Bot Management 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_botmgmt.control	Provide access to all defined operations on the resources tagged as “A” in Figure 1 of section 5.1 in this version of the API.	No
oma_rest_botmgmt.report	Provide access to all defined operations relating to event subscription management (i.e. resources tagged as “B” in Figure 1 of section 5.1) to enable flow of event reports such as spam reports.	No

Table 1: Scope values for RESTful Bot Management API

F.1.1.2 Downscoping

This version of the specification does not define any downscoping values.

F.1.1.3 Mapping with resources and methods

Tables in this section specify how the scope values defined in section F.1.1.1 for the RESTful Bot Management API map to the REST resources and methods of this API. In these tables, the root “oma_rest_botmgmt.” of scope values is omitted for readability reasons.

Resource	URL Base URL: http://{serverRoot}/botmgmt/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
User's anonymization	/{userId}/{chatbotId}/anonymization	6.1	n/a	n/a	control	n/a

Table 2: Required scope values for: Bot Management

Resource	URL Base URL: http://{serverRoot}/botmgmt/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
All subscriptions to bot related notifications	/{botId}/subscriptions	6.3	report	n/a	report	n/a
Individual subscription	/{botId}/subscriptions/{subscriptionId}	6.4	report	n/a	n/a	report

Table 3: Required scope values for: Spam Reports

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.

A client MAY use ‘acr:auth’ in a resource URL in place of a {userId} when the the RESTful Bot Management API is used in combination with [Autho4API_10].

In the case the RESTful Bot Management API supports [Autho4API_10], the server:

- SHALL accept ‘acr:auth’ as a valid value for the resource URL variable {userId}
- SHALL conform to [REST_NetAPI_Common] section 5.8.1.1 regarding the processing of ‘acr:auth’.