RESTful Network API for Notification Channel

Candidate Version 1.0 – 30 Jul 2013

Open Mobile Alliance

OMA-TS-REST_NetAPI_NotificationChannel-V1_0-20130730-C

Contents

1. SCOPE 6

2. REFERENCES 7

2.1 NORMATIVE REFERENCES 7

2.2 INFORMATIVE REFERENCES 7

3. TERMINOLOGY AND CONVENTIONS 8

3.1 CONVENTIONS 8

3.2 DEFINITIONS 8

3.3 ABBREVIATIONS 8

4. INTRODUCTION 10

4.1 VERSION 1.0 10

5. NOTIFICATION CHANNEL API DEFINITION 11

5.1 RESOURCES SUMMARY 12

5.2 DATA TYPES 14

5.2.1 XML Namespaces 14

5.2.2 Structures 14

5.2.2.1 Type: NotificationChannelList 14

5.2.2.2 Type: NotificationChannel 14

5.2.2.3 Type: NotificationList 16

5.2.2.4 Type: ChannelData 16

5.2.2.5 Type: LongPollingData 16

5.2.2.6 Type: OMAPushData 16

5.2.2.7 Type: LongPollingRequestParameters 17

5.2.3 Enumerations 17

5.2.3.1 Enumeration: ChannelType 17

5.2.4 Values of the Link "rel" attribute 17

5.3 SEQUENCE DIAGRAMS 17

5.3.1 Create Notification Channel (Long Polling Method) 18

5.3.2 Notifications delivered to application using Long Polling 19

5.3.3 Long Polling request timeout response 20

5.3.4 Multiple notifications delivered to application in response to the Long Polling request 20

5.3.5 Max number of notifications reached during the Long Polling 21

5.3.6 Create Notification Channel (OMA Push Method) 23

5.3.7 Notifications delivered to application using OMA Push 23

6. DETAILED SPECIFICATION OF THE RESOURCES 25

6.1 RESOURCE: NOTIFICATION CHANNELS 25

6.1.1 Request URL variables 25

6.1.2 Response Codes and Error Handling 26

6.1.3 GET 26

6.1.3.1 Example: Retrieve active Notification Channels (Informative) 26

6.1.3.1.1 Request 26

6.1.3.1.2 Response 26

6.1.4 PUT 27

6.1.5 POST 27

6.1.5.1 Example: Create Notification Channel (Long Polling method), using tel URI (Informative) 27

6.1.5.1.1 Request 27

6.1.5.1.2 Response 27

6.1.5.2 Example: Create Notification Channel (OMA Push method), using tel URI (Informative) 28

6.1.5.2.1 Request 28

6.1.5.2.2 Response 28

6.1.5.3 Example: Create Notification Channel (Long Polling method), using ACR (Informative) 28

6.1.5.3.1 Request 28

6.1.5.3.2 Response 29

6.1.6 DELETE 29

6.2 RESOURCE: INDIVIDUAL NOTIFICATION CHANNEL 29

6.2.1 Request URL variables 29
6.2.2 Response Codes and Error Handling 30
6.2.3 GET 30
6.2.3.1 Example: Retrieve individual Notification Channel (Informative) *30*
6.2.3.1.1 Request 30
6.2.3.1.2 Response 30
6.2.4 PUT 31
6.2.5 POST 31
6.2.6 DELETE 31
6.2.6.1 Example: Removing Notification Channel (Informative) *31*
6.2.6.1.1 Request 31
6.2.6.1.2 Response 31
6.3 RESOURCE: NOTIFICATION LIST 31
6.3.1 Request URL variables 31
6.3.2 Response Codes and Error Handling 31
6.3.3 GET 31
6.3.4 PUT 31
6.3.5 POST 32
6.3.5.1 Example 1: Single notification delivered including content (Informative) *32*
6.3.5.1.1 Request 32
6.3.5.1.2 Response 32
6.3.5.2 Example 2: Multiple notifications delivered including content (Informative) *32*
6.3.5.2.1 Request 32
6.3.5.2.2 Response 33
6.3.5.3 Example 3: Server timeout (Informative) *34*
6.3.5.3.1 Request 34
6.3.5.3.2 Response 34
6.3.6 DELETE 34
7. FAULT DEFINITIONS 35
7.1 SERVICE EXCEPTIONS 35
7.2 POLICY EXCEPTIONS 35
7.2.1 POL1023: OMA Push notification channel not supported 35
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 36
A.1 APPROVED VERSION HISTORY 36
A.2 DRAFT/CANDIDATE VERSION 1.0 HISTORY 36
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 38
B.1 SCR FOR REST.NC SERVER 38
B.1.1 SCR for REST.NC.Channels Server 38
B.1.2 SCR for REST.NC.IndividualChannel Server 38
B.1.3 SCR for REST.NC.LongPolling Server 39
B.1.4 SCR for REST.NC.OMAPush Server 39
APPENDIX C. APPLICATION/X-WWW-FORM-URLENCODED REQUEST FORMAT FOR POST OPERATIONS (NORMATIVE) 40
C.1 CREATING A NOTIFICATION CHANNEL 40
C.1.1 Example 1: Create Notification Channel (Long Polling method), using tel URI (Informative) 41
C.1.1.1 Request *41*
C.1.1.2 Response *41*
C.1.2 Example 2: Create Notification Channel (OMA Push method), using tel URI (Informative) 42
C.1.2.1 Request *42*
C.1.2.2 Response *42*
C.1.3 Example 3: Create Notification Channel, using ACR (Informative) 42
C.1.3.1 Request *42*
C.1.3.2 Response *43*
C.2 RETRIEVING NOTIFICATIONS FROM THE NOTIFICATION SERVER 43
C.2.1 Example 1: Single notification delivered including content (Informative) 43
C.2.1.1 Request *43*
C.2.1.2 Response *44*

APPENDIX D. JSON EXAMPLES (INFORMATIVE) 45
D.1 RETRIEVE ACTIVE NOTIFICATION CHANNELS (SECTION 6.1.3.1) 45
D.2 CREATE NOTIFICATION CHANNEL (LONG POLLING METHOD), USING TEL URI (SECTION 6.1.5.1) 46
D.3 CREATE NOTIFICATION CHANNEL (OMA PUSH METHOD), USING TEL URI (SECTION 6.1.5.2) 47
D.4 CREATE NOTIFICATION CHANNEL (LONG POLLING METHOD), USING ACR (SECTION 6.1.5.3) 47
D.5 CREATE NOTIFICATION CHANNEL (OMA PUSH METHOD), USING ACR (SECTION 6.1.5.4) 48
D.6 RETRIEVE INDIVIDUAL NOTIFICATION CHANNEL (SECTION 6.2.3.1) 49
D.7 REMOVING NOTIFICATION CHANNEL (SECTION 6.2.6.1) 50
D.8 SINGLE NOTIFICATION DELIVERED INCLUDING CONTENT (SECTION 6.3.5.1) 50
D.9 MULTIPLE NOTIFICATIONS DELIVERED INCLUDING CONTENT (SECTION 6.3.5.2) 51
D.10 SERVER TIMEOUT (SECTION 6.3.5.3) 52
APPENDIX E. OPERATIONS MAPPING TO A PRE-EXISTING BASELINE SPECIFICATION (INFORMATIVE) 53
APPENDIX F. LIGHT-WEIGHT RESOURCES (INFORMATIVE) 54
APPENDIX G. AUTHORIZATION ASPECTS (NORMATIVE) 55
G.1 USE WITH OMA AUTHORIZATION FRAMEWORK FOR NETWORK APIS 55
G.1.1 Scope values 55
G.1.1.1 Definitions 55
G.1.1.2 Downscoping 55
G.1.1.3 Mapping with resources and methods 56
G.1.2 Use of 'acr:auth' 58
APPENDIX H. NOTIFICATION SERVER - PUSH ENABLER INTERACTION (INFORMATIVE) 59

Figures

Figure 1 Resource structure defined by this specification 12
Figure 2 Create Notification Channel 18
Figure 3 Notifications delivered to application 19
Figure 4 Request timeout 20
Figure 5 Multiple notifications delivered to application in response 21
Figure 6 Maximum number of notifications in the response to the Long Polling 22
Figure 7 Create Notification Channel (OMA Push Method) 23
Figure 8 Notifications delivered to application using OMA Push 24

Tables

Table 1: Scope values for RESTful Notification Channel API 55
Table 2: Required scope values for: Management of Notification Channel 57
Table 3: Required scope values for: Retrieval of notifications from Notification Server 57

1. Scope

This specification defines a RESTful API for Notification Channel using HTTP protocol bindings.

2. References

2.1 Normative References

[IETF_ACR_draft] "The acr URI for anonymous users", S.Jakobsson, K.Smith, July 2011, URL:http://tools.ietf.org/html/draft-uri-acr-extension-03

[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_Notification Channel] "XML schema for the RESTful Network API for Notification Channel", Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_notificationchannel-V1_0, URL:http://www.openmobilealliance.org/

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

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et. al, January 1999, URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC3261] "SIP: Session Initiation Protocol", J. Rosenberg et al., June 2002, URL:http://www.rfc-editor.org/rfc/rfc3261.txt

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

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

[RFC4627] "The application/json Media Type for JavaScript Object Notation (JSON)", D. Crockford, July 2006, URL:http://www.ietf.org/rfc/rfc4627.txt

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

[W3C_URLENC] HTML 4.01 Specification, Section 17.13.4 Form content types, The World Wide Web Consortium, URL:http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.1

[XMLSchema1] W3C Recommendation, XML Schema Part 1: Structures Second Edition, URL: http://www.w3.org/TR/xmlschema-1/

[XMLSchema2] W3C Recommendation, XML Schema Part 2: Datatypes Second Edition, URL: http://www.w3.org/TR/xmlschema-2/

[OMA_PUSH] "OMA Push 2.3" Open Mobile Alliance™. OMA-ERP-Push-V2_3 URL:http://www.openmobilealliance.org/

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/

[PushOTA] "Push Over-the-Air", Open Mobile Alliance™. OMA-TS-PushOTA-V2_3, URL:http://www.openmobilealliance.org/

[PushPAP] "Push Access Protocol", Open Mobile Alliance™. OMA-TS-PAP-V2_3, URL:http://www.openmobilealliance.org/

[PushREST] "RESTful Network API for OMA Push", Open Mobile Alliance™. OMA-TS-REST_NetAPI_Push-V1_0, 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/

[RFC6202] "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", April 2011, URL:http://tools.ietf.org/rfc/rfc6202.txt

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.
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. 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 callback 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
PAP	Push Access Protocol
PPG	Push Proxy Gateway
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 for the RESTful Network API for Notification Channel contains HTTP protocol bindings for Notification Channel, 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, and application/x-www-form-urlencoded).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations:

- Manage Notification Channel
- Retrieve asynchronous notifications from the Notification Server via Long Polling (i.e. Pull method)
- Receive asynchronous notifications from the Notification Server via OMA Push (i.e Push method)

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 a resource URL variable that identifies an end user

5. Notification Channel API definition

This section is organized to support a comprehensive understanding of the Notification Channel 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 introduces a method for a client (e.g. a browser or a native application) to receive asynchronous notifications from a Notification Server about the events the client has subscribed with one or more Enabler servers. Two notification delivery methods have been specified in this document: Pull and Push methods. The Pull notification delivery method used is based on HTTP requests and often referred as “HTTP Long Polling” [RFC6202]. The Push notification delivery method is based on “OMA Push” [PUSH_ARCH]. This specification assumes the Notification Server, as a Push Initiator, knows how to interact with PPG using Push Access Protocol (PAP) [OMA PUSH] and as such not in the scope of this document.

For both notification delivery methods, as notifications are conveyed through a Notification Channel, the channel must be created first before a Long Polling request can be invoked or an asynchronous event-push can be initiated by the channel onto PPG.

In response to a channel creation request containing channelType = LongPolling, the Notification Server, will provide two URLs: callback URL and channel URL.The client uses callback URL as notification endpoint when subscribing for notifications from the Enabler server(s). Thus, each Enabler server will send subsequent notifications using this callback URL pointing to the Notification Server. The channel URL is used to retrieve notifications from the Notification Server using HTTP Long Polling mechanism. A single Notification Channel may handle notifications from several Enabler servers. Note that the client subscriptions to notifications are specific for each Enabler server and they are not in the scope of this specification.

When the Notification Server receives a notification from an Enabler server, it conveys the notification to the client with the response to the pending HTTP Long Polling request. A Notification Channel has certain time-to-live and therefore in order to continue using it, the channel has to be maintained. With each Long Polling request, the Notification Server will reset the channel life time to its original value.

In response to a channel creation request containing channelType = OMAPush, the Notification Server will only provide a callback URL. That is, for OMAPush notification delivery method, the notification server would not provide a channel URL as the client application is expected to asynchronously receive events via the OMA Push enabler [OMA_PUSH]. As explained earlier above, the client application would use the callback URL as notification endpoint when subscribing to notifications from the Enabler server(s).

Additionally, the request for a channel creation of type OMA Push may contain a unique application Id (appId) which is required by the OMA Push infrastructure [OMA_PUSH] to direct the asynchronous events to a particular client application on the device. However, if the application Id is not present in the channel creation request, it is assumed that the Notification Sever has other means of retrieving the application Id (e.g. through the usage of the available OAuth token in the Notification Channel creation request).

Similarly, an OMA Push notification channel has certain time-to-live and therefore in order to continue using it, the channel has to be maintained. With each event Push, the Notification Server will reset the channel life time to its original value.

It should be noted that in order not to disclose underlying network topology, the Notification Server usually sends to the client a mapped version of the real callback URL. Later, when the Enabler server receives such mapped callback URL, it will apply de-mapping of the URL before it can be used. How this mapping and de-mapping is performed on the server is out of scope for this specification.

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 commands, the possible HTTP response codes, 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. Application/x-www-form-urlencoded examples are provided in Appendix C, while JSON examples are provided in Appendix D.

Section 7 contains fault definition details such as Service Exceptions and Policy Exceptions. Appendix B provides the Static Conformance Requirements (SCR).

Appendix E provides the operations mapping to a pre-existing baseline specification, where applicable.

Appendix F provides a list of all light-weight resources, where applicable.

Appendix G 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 Notification Channel API.

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.

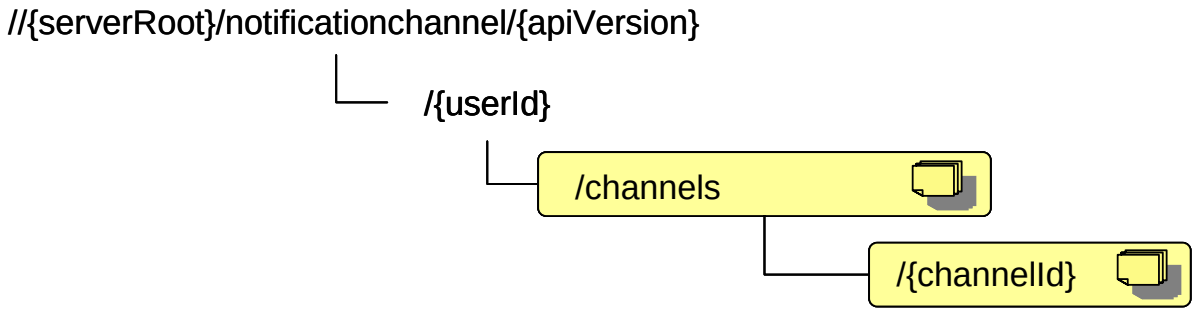

Figure 1 Resource structure defined by this specification

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 the client to manage Notification Channels

Resource	URL Base URL: http://{serverRoot}/notificationchannel/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Notification Channels	/{userId}/channels	NotificationChannelList (used for GET) NotificationChannel (used for POST)	Retrieves a list of Notification Channels.	no	Creates a new Notification Channel.	no
Individual Notification Channel	/{userId}/channels/{channelId}	NotificationChannel (used for GET)	Retrieves an individual Notification Channel.	no	no	Removes an individual Notification Channel.

Purpose: To allow the client to retrieve notifications from the Notification Server by using Long Polling

Resource	URL: < specified by the server >	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Notification list	< Resource URL is received with "channelURL" in response from the server when a Long Polling Notification Channel is created>	LongPollingRequestParameters (used for POST request) NotificationList (used in response to the Long Polling POST request)	no	no	Retrieves pending notifications from the identified Long Polling Notification Channel. At the same time the channel life time is reset to its original value.	no

5.2 Data Types

5.2.1 XML Namespaces

The XML namespace for the Notification Channel data types is:

urn:oma:xml:rest:netapi:notificationchannel: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_NotificationChannel].

5.2.2 Structures

The subsections of this section define the data structures used in the Notification Channel API.

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

5.2.2.1 Type: NotificationChannelList

This type defines a list of Notification Channels.

Element	Type	Optional	Description
notificationChannel	NotificationChannel [0..unbounded]	Yes	May contain an array of Notification Channels.
resourceURL	xsd:anyURI	No	Self referring URL

A root element named notificationChannelList of type NotificationChannelList is allowed in response bodies.

5.2.2.2 Type: NotificationChannel

This type defines a single Notification Channel.

Element	Type	Optional	Description
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 SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate channel creation in such situations. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
applicationTag	xsd:string	Yes	A tag that the client MAY use to tag this particular resource on the server. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.

channelType	ChannelType	No	Specifies the type of Notification Channel to be used: Long Polling or OMA Push (method that will be used to receive new notifications on the channel).
channelData	ChannelData	Yes	Contains specific information for the Notification Channel type specified in channelType. The channelData MUST be included in the response to the request for the creation of Notification Channel for Long Polling or OMA Push. Note that for Long Polling, the channel data is defined in the type LongPollingData (see 5.2.2.5). For OMA Push, the channel data is defined in the type OMAPushData (see 5.2.2.6). Both LongPollingData and OMAPushData are derived from ChannelData. In XML implementation for channelData, LongPollingData or OMAPushData type is identified by the xsi:type attribute.
channelLifetime	xsd:int	Yes	Lifetime (duration) of Notification Channel in seconds. Client can specify desired lifetime of Notification Channel in POST request when creating Notification Channel, however the server in the response to the request may change the desired lifetime according to its server policy. If the element is not present in the POST request, a default channel lifetime specified by server policy will apply. The server SHALL always include the channel lifetime in the response either when it was modified compared to what the client requested, or a default channel lifetime is used.
callbackURL	xsd:anyURI	Yes	Specified by the server. Contains a callback URL used when establishing subscriptions for notifications from the respective Enabler server (not part of this specification). The callbackURL SHALL NOT be included in POST request to create the Notification Channel resource. MUST be included in responses to the channel creation and any HTTP method that returns an entity body.
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 be also included in responses to any HTTP method that returns an entity body, and in PUT requests.

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

5.2.2.3 Type: NotificationList

This type defines a list of notifications that are being delivered to the client.

Element	Type	Optional	Description
<Element is defined by respective Enabler server API>	<Type is defined by respective enabler API> [0..unbounded]	Yes	Contains a list (array) of notifications. The notification types are defined by the different OMA RESTful Network APIs. The list does not impose any further restriction on its content, i.e. notifications of a particular type can occur 0 or more times in the list.

A root element named notificationList of type NotificationList is allowed in response bodies.

5.2.2.4 Type: ChannelData

This is an abstract data type that contains no elements. Data type that is used to define specific information for a particular Notification Channel type (channelData in 5.2.2.2), SHALL be derived from this data type.

5.2.2.5 Type: LongPollingData

This type is derived from ChannelData and it defines specific data for the Long Polling mechanism that is used on the Notification Channel. It is used inside the 'channelData' element when a channel is created, and it is identified by xsi:type attribute.

Element	Type	Optional	Description
channelURL	xsd:anyURI	Yes	Specified by the server. Contains the URL used to retrieve new events. The channelURL SHALL NOT be included in POST request to create the Notification Channel resource, but MUST be included in the response to the channel creation and any HTTP method that returns an entity body.
maxNotifications	xsd:int	Yes	Defines the maximum number of notifications that may be delivered in a notification list. If not specified, a default value specified by the server policy will apply, and the server SHOULD include that value in the response to the client.

5.2.2.6 Type: OMAPushData

This type is derived from ChannelData and it defines specific data for the OMAPush mechanism that is used on the Notification Channel. It is used inside the 'channelData' element when a channel is created, and it is identified by xsi:type attribute.

Element	Type	Optional	Description
appId	xsd:string	Yes	appId is a required data parameter by OMA Push enabler for routing the Push Message to the appropriate application on the target device/MSISDN.
maxNotifications	xsd:int	Yes	Defines the maximum number of notifications that may be delivered in a notification list. Note: the actual deliverable notifications may be limited by the capabilities of the Push-OTA bearer, e.g. up to a particular total size of the notification data. If not specified, a default value specified by the server policy will apply, and the server SHOULD include that

			value in the response to the client.

5.2.2.7 Type: LongPollingRequestParameters

This type defines parameters for Long Polling request.

Element	Type	Optional	Description
(empty)			In the current version of this specifications, this type is empty

A root element named longPollingRequestParameters of type LongPollingRequestParameters is allowed in request bodies.

5.2.3 Enumerations

The subsections of this section define the enumerations used in the Notification Channel API.

5.2.3.1 Enumeration: ChannelType

Enumeration	Description
LongPolling	Indicates that the HTTP Long Polling mechanism is to be used on the Notification Channel to retrieve notifications from the Notification Server.
OMAPush	Indicates that the OMA Push mechanism is to be used by the Notification Server to asynchronously notify the client of events.

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):

- NotificationChannelList
- NotificationChannel
- NotificationList

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.

Note that signalling sequences between the Notification Server and Enabler servers X (e.g. Presence server) and Y (e.g. Messaging server), as well as the signalling sequences between the application and the Enabler servers X and Y (depicted in grey colour) are not part of this specifications; those sequences in the flows are shown for completeness only.

Upon creation of a Notification Channel, the application is required to inform the Notification Server as to the desired notification delivery mechanism. The following two notification delivery mechanisms are supported:

1. Long Polling
2. OMA Push

5.3.1 Create Notification Channel (Long Polling Method)

This figure below shows a scenario for creation of a Notification Channel by an application using the Long Polling notification delivery mechanism. For information on creation of a Notification Channel using OMA Push notification delivery mechanism please refer to section 5.3.6.

The resources:

- To create Notification Channel:
  http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels
- To retrieve new notifications:
  The resource to be used is provided in the response to the channel creation.

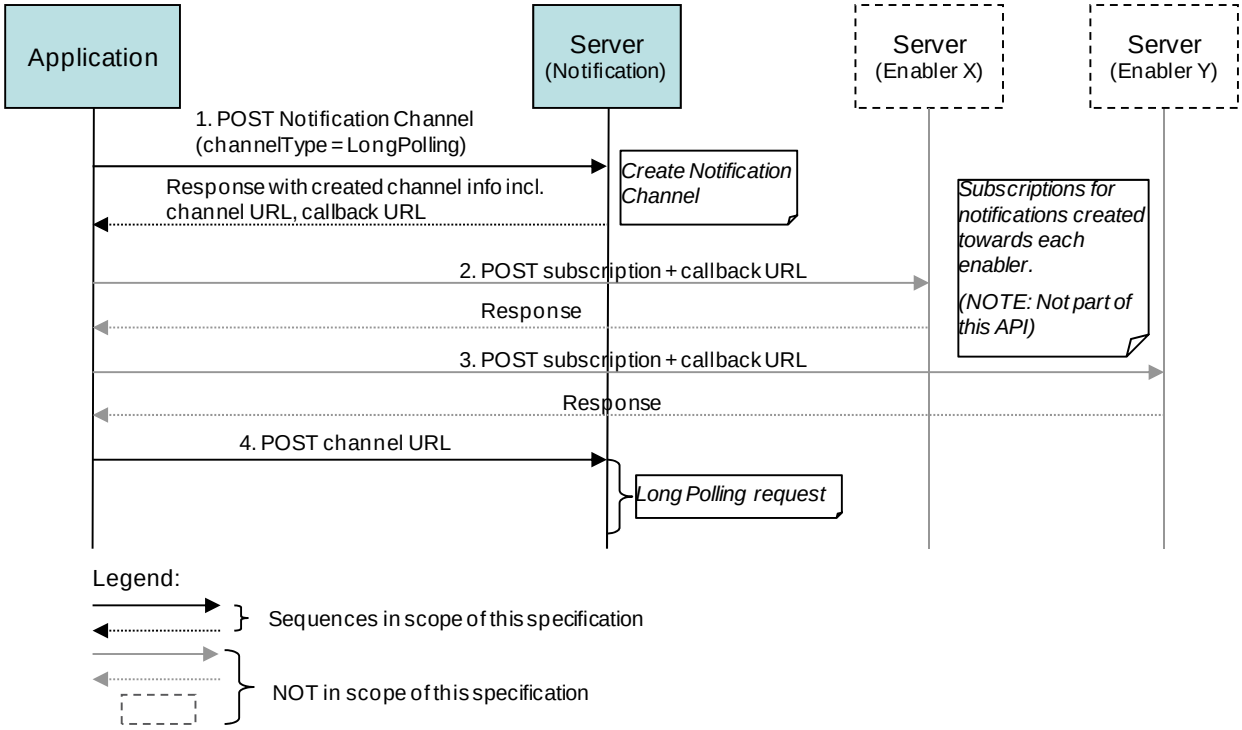

Figure 2 Create Notification Channel

Outline of the flows:

1. Application creates a Notification Channel by sending a POST request to the Notification Server indicating the desire to use the Long Polling notification delivery method by setting the channelType = LongPolling (the request may include a limit to the number of notifications that the application can receive in the responses).

   A successful response includes a body containing a unique channel URL which is to be used when issuing the Long Polling request and callback URL which is to be used when subscribing for notifications to a particular Enabler server.

2. Application creates a subscription for notifications from Enabler X server. The included callback URL instructs the Enabler X server to send notifications to the Notification Server (this operation is not part of this API).

   The Enabler server returns a response (this operation is not part of this API).

3. Application creates a subscription for notifications from Enabler Y server. The included callback URL instructs the Enabler server to send notifications to the Notification Server (this operation is not part of this API).

   The Enabler Y server returns a response.(this operation is not part of this API).

4. Application initiates a Long Polling request using the channel URL received in the response to POST in step 1 and waits for a new event.

5.3.2 Notifications delivered to application using Long Polling

This figure below shows a scenario where two notifications are delivered to the application, generated by two different servers.

The resource used by the application for the Long Polling requests is provided by the Notification Server (e.g. received in the response to creation of the Notification Channel, see section 5.3.1).

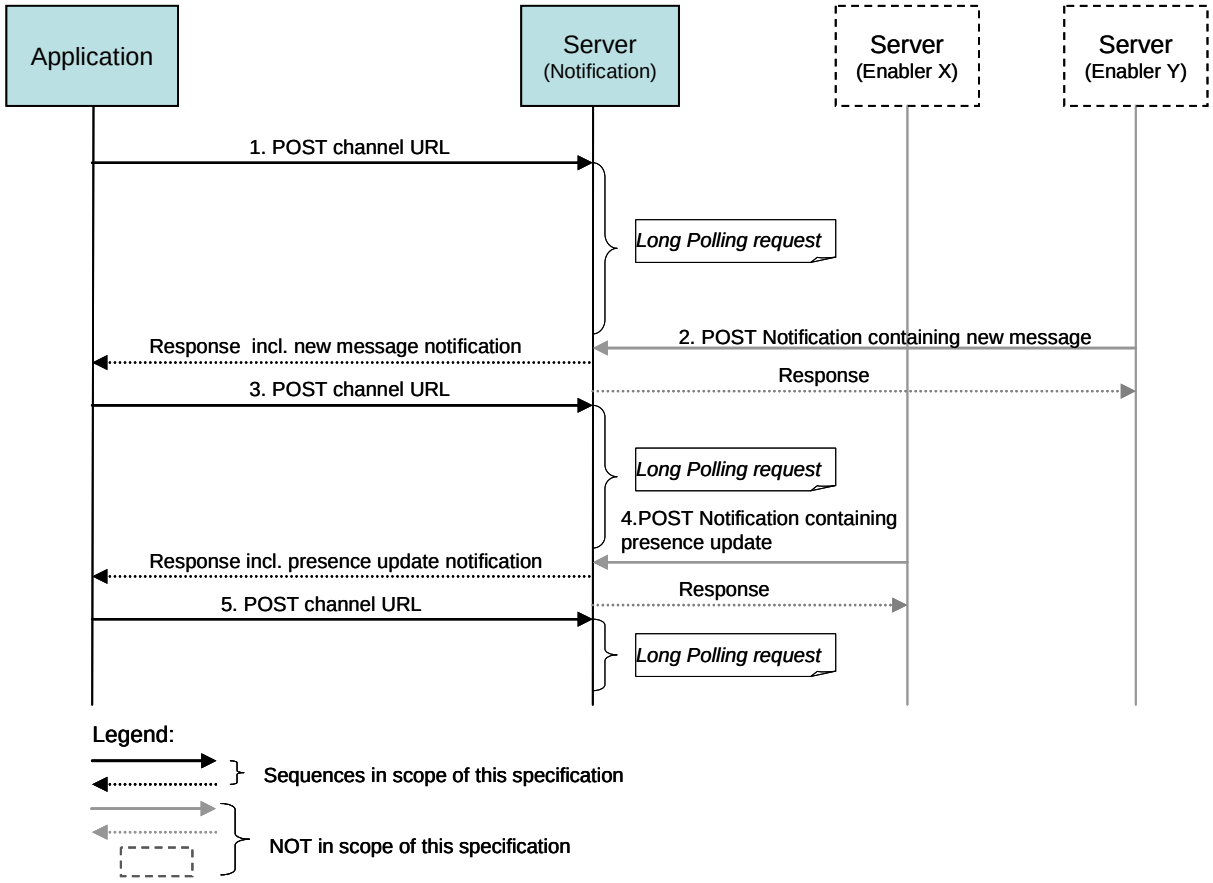

Figure 3 Notifications delivered to application

Outline of the flows:

1. Application initiates a Long Polling request using the channel URL received when the Notification Channel was created.
2. A new message is received, which triggers a notification being sent from the Enabler Y server to the Notification Server using the callback URL provided when the Notification Channel was created (this operation is not part of this API).

   A response to the Long Polling request in step 1 is delivered to the application including the new message.

   A response to the notification received in step 2 is sent to Enabler Y server after the response is delivered to the application (this operation is not part of this API).
3. Application immediately initiates a new Long Polling request.
4. A new event occurs; in this case a presence update notification is received in the Notification Server using the callback URL provided when the Notification Channel was created (this operation is not part of this API).

   A response to the Long Polling request in step 3 is delivered to the application including the presence update.

A response to the notification received in step 4 is sent to Enabler X server after the response is delivered to the application (this operation is not part of this API).

5. Application immediately initiates a new Long Polling request and waits for a new event.

5.3.3 Long Polling request timeout response

This figure below shows a scenario where a Long Polling request times out and a new Long Polling request is sent.

The resource used by the application for the Long Polling requests is provided by the Notification Server (e.g. received in the response to creation of the Notification Channel, see section 5.3.1).

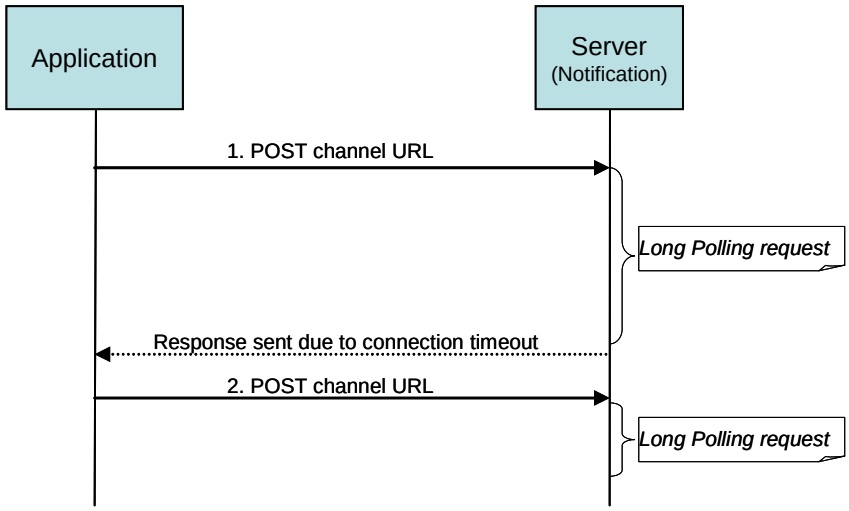

Figure 4 Request timeout

Outline of the flows:

1. Application initiates a Long Polling request using the channel URL received when the Notification Channel was created.

   No new event is received within a given time limit causing the request to timeout. An empty response is returned to the application.

2. Application immediately initiates a new Long Polling request and waits for a new event.

5.3.4 Multiple notifications delivered to application in response to the Long Polling request

This figure below shows a scenario where two notifications are delivered to the application in the same response.

The resource used by the application for the Long Polling requests is provided by the Notification Server (e.g. received in the response to creation of the Notification Channel, see section 5.3.1).

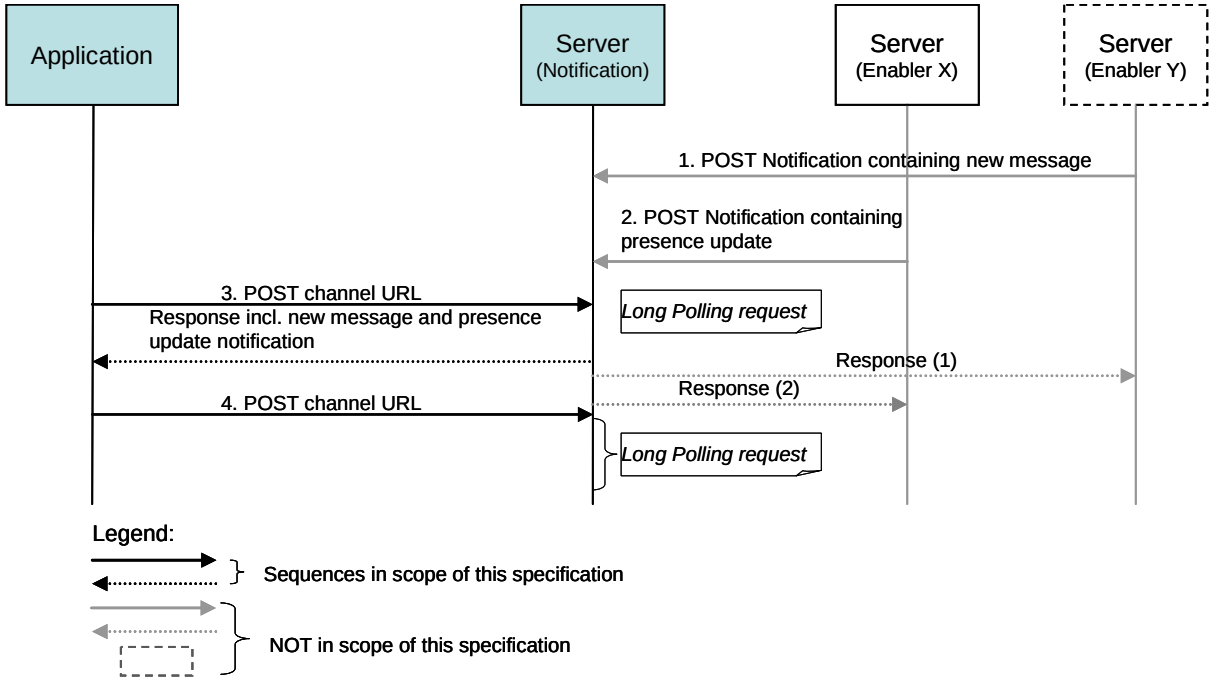

Figure 5 Multiple notifications delivered to application in response

Outline of the flows:

1. A new message is received but in this case there is no outstanding Long Polling request from the application so the notification will be pending in the Notification Server (this operation is not part of this API).
2. A new event occurs; in this case a presence update notification is received. As there is no outstanding Long Polling request from the application the notification will be pending in the Notification Server (this operation is not part of this API).
3. Application initiates a Long Polling request using the channel URL received when the Notification Channel was created.

   A response to the Long Polling request in step 3 is delivered to the application including the new message and the presence update notification (assuming that the application allowed multiple notifications in the response when the Notification Channel was created).

   A response to the notification received in step 1 is sent to Enabler Y server after the response is delivered to the application (this operation is not part of this API).

   A response to the notification received in step 2 is sent to Enabler X server after the response is delivered to the application (this operation is not part of this API).
4. Application immediately initiates a new Long Polling request and waits for a new event.

5.3.5 Max number of notifications reached during the Long Polling

This figure below shows a scenario where the limit for the number of notifications in the response to the application (in this example, 3 notifications) has been reached, which triggered response back to the application.

The resource used by the application for the Long Polling requests is provided by the Notification Server (e.g. received in the response to creation of the Notification Channel, see section 5.3.1).

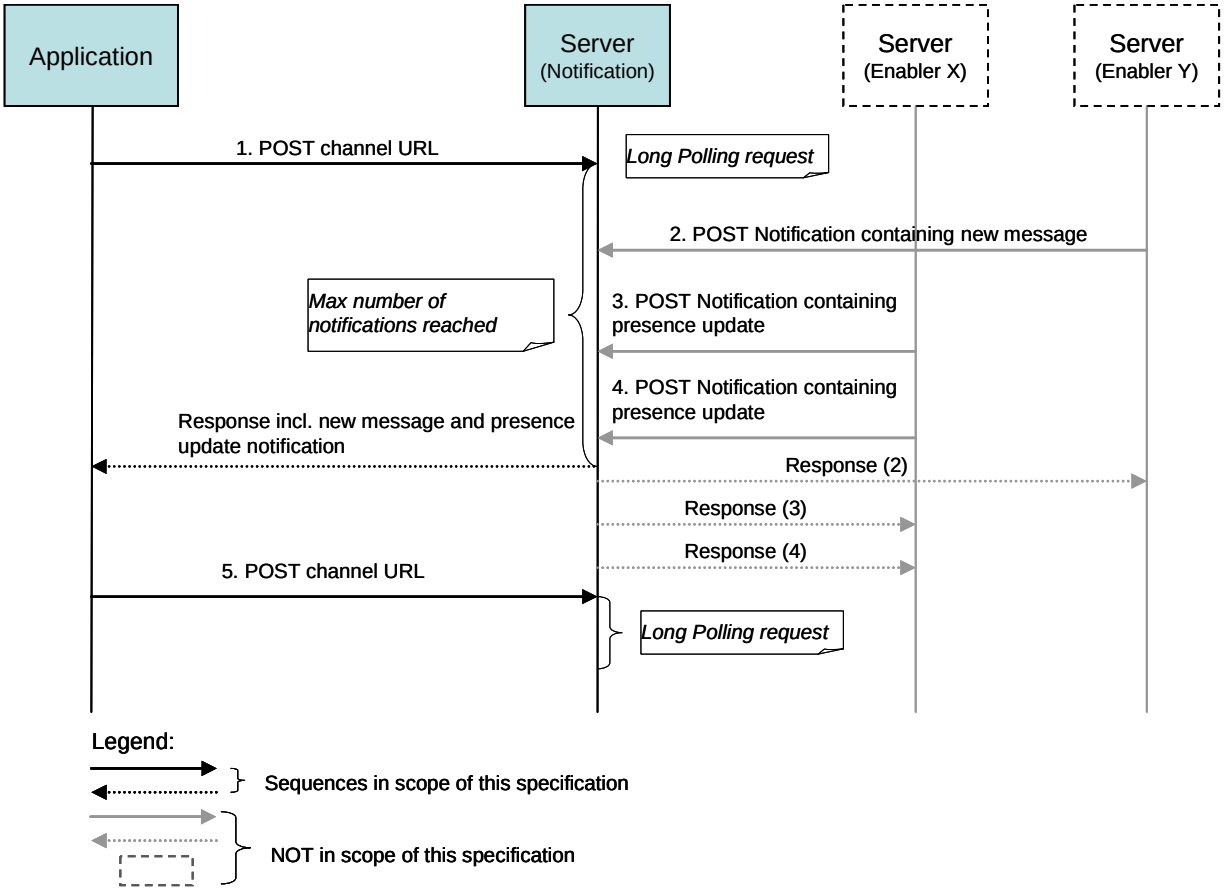

Figure 6 Maximum number of notifications in the response to the Long Polling

Outline of the flows:

1. Application initiates a Long Polling request using the channel URL received when the Notification Channel was created.
2. A new message has been received and the Notification Server is notified (this operation is not part of this API). Since the maxNotifications limit is not yet reached no response to the Long Polling request is sent back to the application.
3. A new event occurs; in this case a presence update notification is received at the Notification Server (this operation is not part of this API). The maxNotifications limit is still not reached.
4. A new event occurs; in this case another presence update notification is received at the Notification Server (this operation is not part of this API).

   The maximum number of notifications allowed in the response has been reached and the response to the Long Polling request in step 1 is sent to the application. The response includes the new message and the two presence update notifications.

   A response to the notification received in step 2 is sent to Enabler Y server after the response is delivered to the application (this operation is not part of this API).

   A response to the notification received in step 3 is sent to Enabler X server after the response is delivered to the application (this operation is not part of this API).

   A response to the notification received in step 4 is sent to Enabler X server after the response is delivered to the application (this operation is not part of this API).
5. Application immediately initiates a new Long Polling request.

5.3.6 Create Notification Channel (OMA Push Method)

This figure below shows a scenario for creation of a Notification Channel by an application using the OMA Push notification delivery mechanism.

The resources:

- To create Notification Channel:
  http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels

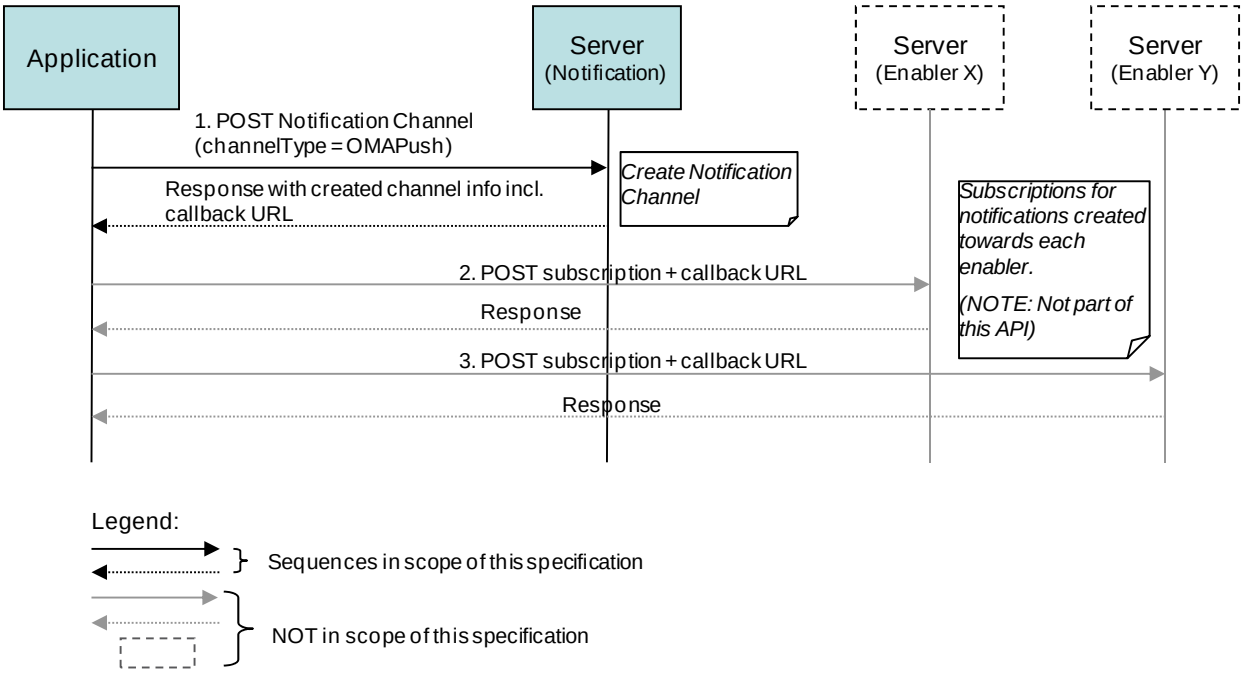

Figure 7 Create Notification Channel (OMA Push Method)

Outline of the flows:

1. Application creates a Notification Channel by sending a POST request to the Notification Server indicating the desire to use the OMA Push notification delivery method by setting the channelType = OMAPush. The request may include a limit to the number of notifications that the application can receive in the asynchronous notification list. Additionally, the request may contain an appId which uniquely identify the application to the OMA Push Enabler.

   A successful response includes a body containing a callback URL which is to be used when subscribing for notifications to a particular Enabler server.

2. Application creates a subscription for notifications from Enabler X server. The included callback URL instructs the Enabler X server to send notifications to the Notification Server (this operation is not part of this API).

   The Enabler server returns a response (this operation is not part of this API).

3. Application creates a subscription for notifications from Enabler Y server. The included callback URL instructs the Enabler server to send notifications to the Notification Server (this operation is not part of this API).

   The Enabler Y server returns a response.(this operation is not part of this API).

5.3.7 Notifications delivered to application using OMA Push

This figure below shows a scenario where two notifications generated by two different servers are delivered to the application,

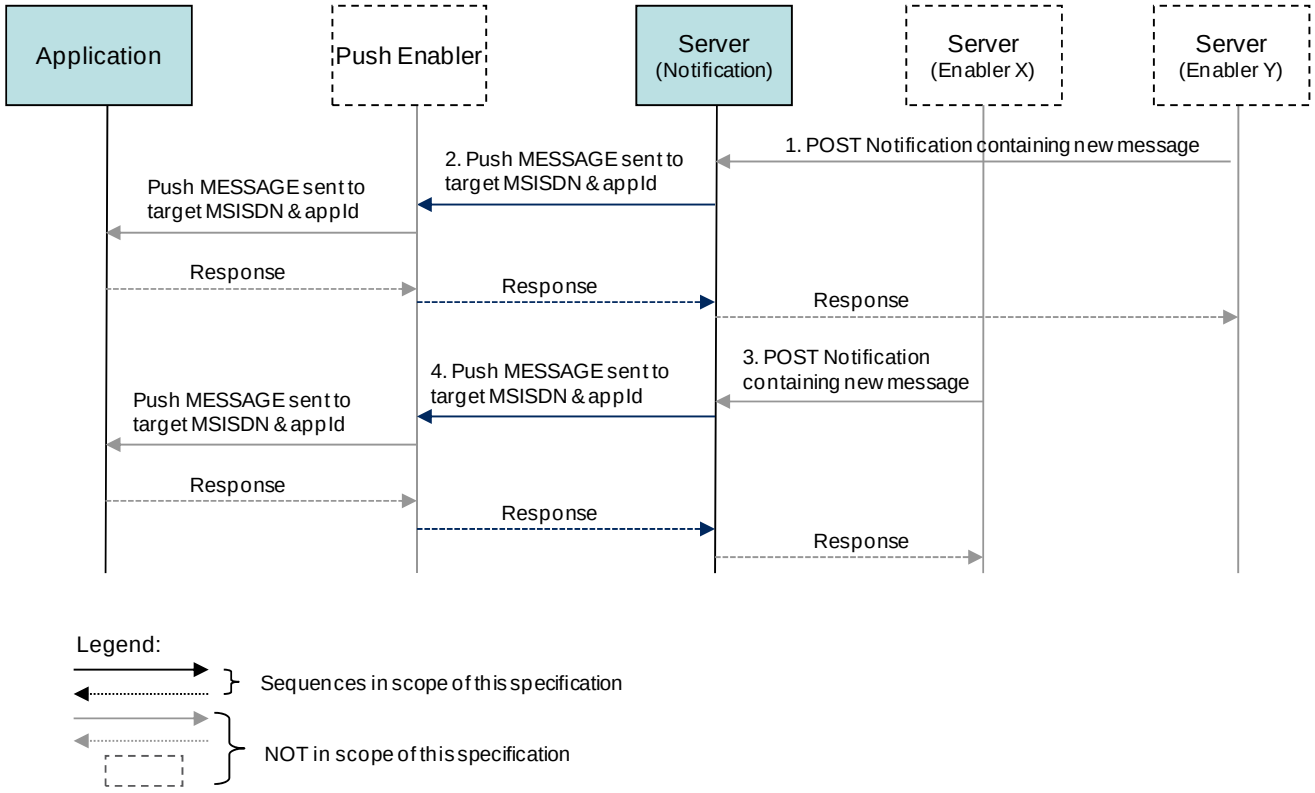

Figure 8 Notifications delivered to application using OMA Push

Outline of the flows:

1. An event occurs which triggers a notification being sent from Enabler Y server to the Notification Server using the callback URL provided when the Notification Channel was created (this operation is not part of this API).
2. The Notification Server maps the callback URL at which it received the event to the associated MSISDN and appId which it had previously captured as part of the channel creation process . APush MESSAGE containing the new event is then sent from the Notification Server to the Push Enabler targeting the appropriate MSISDN and appId (this operation is not part of this API See Appendix H for further information regarding Notification Server and Push Enabler interaction).

   Note: In advance configuration of the Notification Server with the appropriate Push Enabler (e.g. PPG) address is outside the scope of this document.

   In turn, Push Enabler passes the Push MESSAGE containing the new event to the application on the device via the Push client residing on the device (this operation is not part of this API).

   If requested by the Notification Server, the Push client or application may provide a delivery confirmation, which is forwarded to the Notification Server by the Push Enabler (this operation is not part of this API).

   A response to the notification received in step 1 is sent to Enabler Y server after the response is delivered to the application (this operation is not part of this API).
3. The same process as explain in step 1 above involving Enabler X.
4. The same process as explain in step 2 above involving Enabler X.

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, application/x-www-form-urlencoded):

- 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, userId, 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 a user identifier (e.g. address, userId, etc) 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 [IETF_ACR_draft], 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 G.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, and MAY support application/x-www-form-urlencoded 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: Notification channels

The resource used is:

http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels

This resource is used for create a new Notification Channel as well as to obtain a list of active Notification Channels for the specified user.

6.1.1 Request URL variables

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

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	User identifier. Examples: tel:+19585550100, acr:pseudonym123

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 Notification Channel, see section 7.

6.1.3 GET

This operation is used for retrieval of active Notification Channels.

6.1.3.1 Example: Retrieve active Notification Channels (Informative)

6.1.3.1.1 Request

```
GET /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Accept: application/xml
```

6.1.3.1.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannelList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<notificationChannel>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
     <channelURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications
     </channelURL>
     <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu111</callbackURL>
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123</resourceURL>
</notificationChannel>
<notificationChannel>
  <clientCorrelator>987</clientCorrelator>
  <applicationTag>someOtherApp</applicationTag>
  <channelType>OMAPush</channelType>
  <channelData xsi:type="nc:OMAPushData">
     <appId>x-wap-application:wml.ua</appId>
     <maxNotifications>5</maxNotifications>
  </channelData>
  <channelLifetime>3600</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu222</callbackURL>
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987</resourceURL>
</notificationChannel>
<resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels</resourceURL>
</nc:notificationChannelList>
```

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 section 14.7 of [RFC2616].

6.1.5 POST

This operation is used for creation of a Notification Channel in order to receive notifications from an Enabler server to which the client has subscribed for notifications.

6.1.5.1 Example: Create Notification Channel (Long Polling method), using tel URI (Informative)

6.1.5.1.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
     <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
</nc:notificationChannel>
```

6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<clientCorrelator>123</clientCorrelator>
<applicationTag>myApp</applicationTag>
<channelType>LongPolling</channelType>
<channelData xsi:type="nc:LongPollingData">
   <channelURL> http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications
   </channelURL>
   <maxNotifications>1</maxNotifications>
</channelData>
<channelLifetime>7200</channelLifetime>
<callbackURL>http://example.com/callBackUrl/cbu111</callbackURL>
<resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123</resourceURL>
</nc:notificationChannel>
```

6.1.5.2 Example: Create Notification Channel (OMA Push method), using tel URI (Informative)

6.1.5.2.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <clientCorrelator>987</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>OMAPush</channelType>
  <channelData xsi:type="nc:OMAPushData">
     <appId>x-wap-application:wml.ua</appId>
     <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
</nc:notificationChannel>
```

6.1.5.2.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<clientCorrelator>987</clientCorrelator>
<applicationTag>myApp</applicationTag>
<channelType>OMAPush</channelType>
<channelData xsi:type="nc:OMAPushData">
   <appId>x-wap-application:wml.ua</appId>
   <maxNotifications>1</maxNotifications>
</channelData>
<channelLifetime>7200</channelLifetime>
<callbackURL>http://example.com/callBackUrl/cbu222</callbackURL>
<resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987</resourceURL>
</nc:notificationChannel>
```

6.1.5.3 Example: Create Notification Channel (Long Polling method), using ACR (Informative)

6.1.5.3.1 Request

```
POST /exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
```



```
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
     <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
</nc:notificationChannel>
```

6.1.5.3.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<clientCorrelator>123</clientCorrelator>
<applicationTag>myApp</applicationTag>
<channelType>LongPolling</channelType>
<channelData xsi:type="nc:LongPollingData">
   <channelURL> http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123/notifications
   </channelURL>
   <maxNotifications>1</maxNotifications>
</channelData>
<channelLifetime>7200</channelLifetime>
<callbackURL>http://example.com/callBackUrl/cbu111</callbackURL>
<resourceURL>http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123</resourceURL>
</nc:notificationChannel>
```

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 section 14.7 of [RFC2616].

6.2 Resource: Individual Notification Channel

The resource used is:

http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels/**{channelId}**

This resource is used for management of an individual Notification Channel, operations such as: to retrieve information of the Notification Channel or to remove (terminate) Notification Channel.

6.2.1 Request URL variables

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

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	User identifier. Examples: tel:+19585550100, acr:pseudonym123
channelId	Channel identifier. Example: ch456

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 Notification Channel, see section 7.

6.2.3 GET

This operation is used for retrieval of an individual Notification Channel.

6.2.3.1 Example: Retrieve individual Notification Channel (Informative)

6.2.3.1.1 Request

```
GET /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456 HTTP/1.1
Host: example.com
Accept: application/xml
```

6.2.3.1.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <clientCorrelator>456</clientCorrelator>
  <applicationTag>someOtherApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
    <channelURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456/notifications
    </channelURL>
    <maxNotifications>5</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu333</callbackURL>
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456</resourceURL>
</nc:notificationChannel>
```

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, DELETE' field in the response as per section 14.7 of [RFC2616].

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, DELETE' field in the response as per section 14.7 of [RFC2616].

6.2.6 DELETE

This operation is used for removing an individual Notification Channel. Any outstanding poll request will immediately be responded with a 404 Not Found.

6.2.6.1 Example: Removing Notification Channel (Informative)

6.2.6.1.1 Request

```
DELETE /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456 HTTP/1.1
Host: example.com
```

6.2.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2009 02:51:59 GMT
```

6.3 Resource: Notification list

The resource URL is provided by the server (channel URL received when the Long Polling Notification Channel is created) and therefore this specification does not make any assumption about the structure of this URL.

For the Long Polling method, this resource is used for retrieval of new notifications from the Notification Server, for which the application has subscribed from the respective Enabler server. At the same time, the server resets the channel lifetime to its original value.

6.3.1 Request URL variables

Provided by the Notification Server in response to request for creation of a Long Polling Notification Channel.

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 Notification Channel, see section 7.

6.3.3 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 section 14.7 of [RFC2616].

6.3.4 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 section 14.7 of [RFC2616].

6.3.5 POST

This operation is used for retrieval of new notifications from the Notification Server if the Notification Channel involved is of Long Polling type.

6.3.5.1 Example 1: Single notification delivered including content (Informative)

In this example a presence update is delivered to the application.

6.3.5.1.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:longPollingRequestParameters xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1"/>
```

6.3.5.1.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1">
  <pr:presenceNotification xmlns:pr="urn:oma:xml:rest:presence:1">
    <presentityUserId>tel:+19585550100</presentityUserId>
    <callbackData>1234</callbackData>
    <resourceStatus>Active</resourceStatus>
    <presence>
      <person>
        <mood>
          <moodValue>Happy</moodValue>
        </mood>
      </person>
    </presence>
    <link rel="PresenceSubscription"
     href="http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
    tel%3A%2B19585550100/sub001"/>
  </pr:presenceNotification>
</nc:notificationList>
```

6.3.5.2 Example 2: Multiple notifications delivered including content (Informative)

In this example a presence update and message notification are delivered to the application.

6.3.5.2.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
```

```
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:longPollingRequestParameters xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1"/>
```

6.3.5.2.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1">
  <pr:presenceNotification xmlns:pr="urn:oma:xml:rest:presence:1">
    <presentityUserId>tel:+19585550100</presentityUserId>
    <callbackData>1234</callbackData>
    <resourceStatus>Active</resourceStatus>
    <presence>
      <person>
        <mood>
          <moodValue>Happy</moodValue>
        </mood>
      </person>
    </presence>
    <link rel="PresenceSubscription"
href="http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
       tel%3A%2B19585550100/sub001"/>
</pr:presenceNotification>
<mms:inboundMessageNotification xmlns:mms="urn:oma:xml:rest:messaging:1">
    <inboundMessage>
      <destinationAddress>tel:+19585550100</destinationAddress>
      <senderAddress>tel:+19585550101</senderAddress>
      <resourceURL>http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg123
      </resourceURL>
      <link rel="Subscription" href="http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123"/>
      <messageId>msg123</messageId>
      <inboundMMSMessage>
        <subject>Who is RESTing on the beach?</subject>
      </inboundMMSMessage>
    </inboundMessage>
</mms:inboundMessageNotification>
 <mms:inboundMessageNotification xmlns:mms="urn:oma:xml:rest:messaging:1">
    <inboundMessage>
      <destinationAddress>tel:+19585550100</destinationAddress>
      <senderAddress>tel:+19585550102</senderAddress>
      <resourceURL>http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg1234
      </resourceURL>
      <link rel="Subscription" href="http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123"/>
      <messageId>msg1234</messageId>
      <inboundMMSMessage>
        <subject>Who is still RESTing on the beach?</subject>
      </inboundMMSMessage>
    </inboundMessage>
</mms:inboundMessageNotification>
```



```
</nc:notificationList>
```

6.3.5.3 Example 3: Server timeout (Informative)

In this example a Long Polling request times out in the Notification Server before any new notifications from Enabler servers have been received on the server. The server responds with an empty response.

6.3.5.3.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:longPollingRequestParameters xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1"/>
```

6.3.5.3.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1"/>
```

6.3.6 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 section 14.7 of [RFC2616].

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 Notification Channel API.

7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common].

The following additional Policy Exception codes are defined for the Notification Channel API.

7.2.1 POL1023: OMA Push notification channel not supported

Name	Description
MessageID	POL1023
Text	OMA Push notification channel not supported.
Variables	None
HTTP status code(s)	403 Forbidden

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: OMA-TS-REST_NetAPI_NotificationChannel-V1_0	28 Apr 2011	Many	This is the first version of the document that is based on agreed contribution OMA-ARC-RC-APIs-2011-0040R03-INP_Proposal_for_Notification_Channel_TS. In addition, the document title is updated to address the issues from ARC-2011-A071.
	25 May 2011	Many	Implemented CR, OMA-ARC-REST-NetAPI-2011-0008-CR_TS_changes_for_NotificationChannel
	02 Jul 2011	Many	Implemented CRs: OMA-ARC-REST-NetAPI-2011-0092-CR _TS_NotificationChannel_alignment_with_new_template OMA-ARC-REST-NetAPI-2011-0096-CR _TS_NotificationChannel_channelData_type
	25 Jul 2011	Many	Implemented CRs: OMA-ARC-REST-NetAPI-2011-0135R01-CR _TS_NC_duration_timer OMA-ARC-REST-NetAPI-2011-0137-CR _TS_NC_Additional_SCRs OMA-ARC-REST-NetAPI-2011-0147R01-CR _TS_NC_Appendix_C_and_D
	08 Sep 2011	Many	Implemented CRs: OMA-ARC-REST-NetAPI-2011-0210R01-CR_NC_XML_examples_for_channel_duration OMA-ARC-REST-NetAPI-2011-0224-CR_NC_telURI_resourceURL_changes
	21 Sep 2011	Many	Implemented CRs: OMA-ARC-REST-NetAPI-2011-0241-CR _NC_TS_ACR_changes OMA-ARC-REST-NetAPI-2011-0253R02-CR_ NC_TS_clarifications_and_tidy_ups
	03 Nov 2011	Many	Implemented CR: OMA-ARC-REST-NetAPI-2011-0330R02-CR_NC_TS_CONRR_fixing_editorial_comments
	20 Dec 2011	Many	Implemented CRs: OMA-ARC-REST-NetAPI-2011-0451-CR_NC_TS_CONRR_technical_comments_resolution OMA-ARC-REST-NetAPI-2011-0454-CR_NC_TS_Appendix_G
Candidate Version: OMA-TS-REST_NetAPI_NotificationChannel-V1_0	17 Jan 2012	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2012-0007-INP_REST_NetAPI_NotificationChannel_1_0_ERP_and_ETR_for_ Candidate_Approval
Draft Versions: OMA-TS-REST_NetAPI_NotificationChannel-V1_0	24 Jul 2012	5, 6.1.2, 6.2.2, 6.3.2, 7, G.1.1.3	Incorporated CR: OMA-TS-REST_NetAPI_NotificationChannel-V1_0-20120117-C_changes_CR0162 Editorial changes
	24 Aug 2012	5.2.2.2, C.1	Incorporated CR: OMA-ARC-REST-NetAPI-2012-0233-CR_NC_TS_issue_20_clientCorrelator_resolution Editorial changes

Document Identifier	Date	Sections	Description
	15 Oct 2012	2.1, 2.2, 3.3, 4.1, 5, 5.1, 5.2.2.2, 5.2.2.6, 5.2.3.1, 5.3, 5.3.1, 5.3.5, 5.3.6, 5.3.7, 6.1.3.1.2, 6.1.5.1, 6.1.5.1.2, 6.1.5.2, 6.1.5.3, 6.1.5.3.2, 6.3, 6.3.1, 6.3.5, 7.2, B.1.1, B.1.4, C.1.1, C.1.2, C.1.3, C.2, D.1, D.2, D.3, D.4,, D.5, G.1.1.1, G.1.1.2, G.1.1.3	Incorporated CR: OMA-ARC-REST-NetAPI-2012-0254R01-CR_Notification_Channel_support_for_OMA_Push Editorial changes
	08 Nov 2012	2.2, 3.2, 5.1, 5.2.2, 5.2.3.1, 6.1.1, 6.2.1, 7.2.1, B.1, B.1.1, B.1.2, B.1.3, B.1.4, C.1, C.2, D.9, F	Incorporated CR: OMA-ARC-REST-NetAPI-2012-0273R01-CR_NC_TS_NotificationList_fixing_element_description Editorial changes
	19 Nov 2012	6.3.5.2.2, D.9	Incorporated CR: OMA-ARC-REST-NetAPI-2012-0276-CR_Notification_Channel_fixing_and_extending_examples
	13 Dec 2012	4.1, 6, B, G.1.1.1, G.1.1.3, G.1.2,	Incorporated CR: OMA-ARC-REST-NetAPI-2012-0291-CR_NC_TS_implementing_blueprint_for_authorization Template changed to OMA-TEMPLATE-TS_RESTful_Network_API-20120813-I Editorial changes
	15 Apr 2013	2.1, 2.2, 5.3.7, 7.2.1, H	Incorporated CR: OMA-ARC-REST-NetAPI-2013-0019R01-CR_Notification_Server_Push_Enabler_interaction_info Editorial changes
	15 Jul 2013	H	Incorporated CR: OMA-ARC-REST-NetAPI-2013-0048-CR_NotifChannel_fixing_xml_example_for_push_pap Editorial changes
Candidate Version: OMA-TS-REST_NetAPI_NotificationChannel-V1_0	30 Jul 2013	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2013-0224-INP_REST_NetAPI_NotificationChannel_V1_0_ERP_for_Candidate_re_approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for REST.NC Server

Item	Function	Reference	Requirement
REST-NC-SUPPORT-S-001-M	Support for the RESTful Notification Channel API	5, 6	
REST-NC-SUPPORT-S-002-M	Support for the XML request & response format	6	
REST-NC-SUPPORT-S-003-M	Support for the JSON request & response format	6	
REST-NC-SUPPORT-S-004-O	Support for the application/x-www-form-urlencoded format	Appendix C	

B.1.1 SCR for REST.NC.Channels Server

Item	Function	Reference	Requirement
REST-NC-CHANNELS-S-001-M	Support for management of Notification Channels	6.1	
REST-NC-CHANNELS-S-002-O	Retrieving a list of Notification Channels - GET	6.1.3	
REST-NC-CHANNELS-S-003-M	Creating a Long Polling Notification Channel – POST (XML or JSON)	6.1.5	
REST-NC-CHANNELS-S-004-M	Creating a OMA Push Notification Channel – POST (XML or JSON)	6.1.5	
REST-NC-CHANNELS-S-005-O	Creating a Notification Channel – POST (application/x-www-form-urlencoded)	C.1	

B.1.2 SCR for REST.NC.IndividualChannel Server

Item	Function	Reference	Requirement
REST-NC-INDCHANNEL-S-001-M	Support for access to individual Notification Channel	6.2	
REST-NC-INDCHANNEL-S-002-M	Retrieving Notification Channel information - GET	6.2.3	
REST-NC-INDCHANNEL-S-003-M	Terminating Notification Channel – DELETE	6.2.6	

B.1.3 SCR for REST.NC.LongPolling Server

Item	Function	Reference	Requirement
REST-NC-LONGPOLL-S-001-M	Support for access to notifications	6.3	
REST-NC-LONGPOLL-S-002-M	Retrieving notifications from the server using Long Polling - POST	6.3.5	
REST-NC-LONGPOLL-S-003-O	Retrieving notifications from the server using Long Polling – POST (application/x-www-form-urlencoded)	C.2	

B.1.4 SCR for REST.NC.OMAPush Server

Item	Function	Reference	Requirement
REST-NC-OMAPUSH-S-001-M	Acting as a Push Initiator by pushing notifications to OMA Push Enabler	-	

Appendix C. Application/x-www-form-urlencoded Request Format for POST Operations (Normative)

This section defines a format for the RESTful Notification Channel REST API requests where the body of the request is encoded using the application/x-www-form-urlencoded MIME type.

Note: only the request body is encoded as application/x-www-form-urlencoded, the response is still encoded as XML or JSON depending on the preference of the client and the capabilities of the server. Names and values MUST follow the application/x-www-formurlencoded character escaping rules from [W3C_URLENC].

The encoding is defined below for the following Notification Channel REST operations which are based on POST requests:

- Create a Notification Channel
- Retrieve notifications from Notification Server

C.1 Creating a Notification Channel

This operation is used to create a Notification Channel, see section 6.1.5.

The request parameters are as follows:

Name	Type/Values	Optional	Description
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 SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate channel creation in such situations. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
applicationTag	xsd:string	Yes	A tag that the client MAY use to tag this particular resource on the server. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
channelType	xsd:string	No	Specifies the type of Notification Channel to be used (method that will be used to receive new notifications on the channel). Allowed string values are defined in 5.2.3.1.
maxNotifications	xsd:int	Yes	Defines the maximum number of notifications that may be delivered in a notification list. If not specified, a default value specified by the server policy will apply, and the server SHOULD include that value in the response to the client.
channelLifetime	xsd:int	Yes	Lifetime (duration) of Notification Channel in seconds.

			Client can specify desired lifetime of Notification Channel in POST request when creating Notification Channel, however the server in the response to the request may change the desired lifetime according to its server policy. If the element is not present in the POST request, a default channel lifetime specified by server policy will apply. The server SHALL always include the channe lifetime in the response either when it was modified compared to what the client requested, or a default channel lifetime is used.

C.1.1 Example 1: Create Notification Channel (Long Polling method), using tel URI (Informative)

C.1.1.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml

clientCorrelator=123&
applicationTag=myApp&
channelType=LongPolling&
maxNotifications=1&
channelLifetime=7200
```

C.1.1.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
    <channelURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications
    </channelURL>
    <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu111</callbackURL>
```

```
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123</resourceURL>
</nc:notificationChannel>
```

C.1.2 Example 2: Create Notification Channel (OMA Push method), using tel URI (Informative)

C.1.2.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml

clientCorrelator=987&
applicationTag=myApp&
channelType=OMAPush&
appId=x-wap-application:wml.ua&
maxNotifications=1&
channelLifetime=7200
```

C.1.2.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" >
  <clientCorrelator>987</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>OMAPush</channelType>
  <channelData xsi:type="nc:OMAPushData">
    <appId>x-wap-application:wml.ua</appId>
    <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu222</callbackURL>
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987</resourceURL>
</nc:notificationChannel>
```

C.1.3 Example 3: Create Notification Channel, using ACR (Informative)

C.1.3.1 Request

```
POST /exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels HTTP/1.1
Host: example.com
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml

clientCorrelator=123&
applicationTag=myApp&
channelType=LongPolling&
maxNotifications=1&
channelLifetime=7200
```

C.1.3.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationChannel xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" >
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <channelType>LongPolling</channelType>
  <channelData xsi:type="nc:LongPollingData">
    <channelURL>http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123/notifications
    </channelURL>
    <maxNotifications>1</maxNotifications>
  </channelData>
  <channelLifetime>7200</channelLifetime>
  <callbackURL>http://example.com/callBackUrl/cbu111</callbackURL>
  <resourceURL>http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123</resourceURL>
</nc:notificationChannel>
```

C.2 Retrieving notifications from the Notification Server

This operation is used to retrieve new notifications from the Notification Server if the Notification Channel involved is of Long Polling type, see section 6.3.5.

The request parameters are as follows:

Name	Type/Values	Optional	Description
longPollingRequestParameters	(empty)	No	Provides the body of the request, which is an empty string in this version of specification.

C.2.1 Example 1: Single notification delivered including content (Informative)

C.2.1.1 Request

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
```

```
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml

longPollingRequestParmeters=
```

C.2.1.2 Response

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

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1">
  <pr:presenceNotification xmlns:pr="urn:oma:xml:rest:presence:1">
    <presentityUserId>tel:+19585550100</presentityUserId>
    <callbackData>1234</callbackData>
    <resourceStatus>Active</resourceStatus>
    <presence>
      <person>
        <mood>
          <moodValue>Happy</moodValue>
        </mood>
      </person>
    </presence>
    <link rel="PresenceSubscription"
     href="http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
    tel%3A%2B19585550100/sub001"/>
  </pr:presenceNotification>
</nc:notificationList>
```

Appendix D. JSON examples (Informative)

JSON (JavaScript Object Notation) is a lightweight, 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 [RFC4627].

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.

D.1 Retrieve active Notification Channels (section 6.1.3.1)

Request:

```
GET /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

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

{"notificationChannelList": {
   "notificationChannel": [
      {
         "applicationTag": "myApp",
         "callbackURL": "http://example.com/callBackUrl/cbu111",
         "channelData": {
            "channelURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications
",
            "maxNotifications": "1",
            "type": "nc:LongPollingData"
         },
         "channelLifetime": "7200",
         "channelType": "LongPolling",
         "clientCorrelator": "123",
         "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123"
      },
      {
         "applicationTag": "someOtherApp",
         "callbackURL": "http://example.com/callBackUrl/cbu222",
         "channelData": {

            "appId": "x-wap-application:wml.ua",          "maxNotifications": "5",
            "type": "nc:OMAPushData"
         },
         "channelLifetime": "3600",
         "channelType": "OMAPush",
         "clientCorrelator": "987",
```

```
            "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987"
        }
    ],
    "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels"
}}
```

D.2 Create Notification Channel (Long Polling method), using tel URI (section 6.1.5.1)

Request:

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: nnnn
Accept: application/json

{"notificationChannel": {
    "applicationTag": "myApp",
    "channelData": {
        "maxNotifications": "1",
        "type": "nc:LongPollingData"
    },
    "channelLifetime": "7200",
    "channelType": "LongPolling",
    "clientCorrelator": "123"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"notificationChannel": {
    "applicationTag": "myApp",
    "callbackURL": "http://example.com/callBackUrl/cbu111",
    "channelData": {
        "channelURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications",
        "maxNotifications": "1",
        "type": "nc:LongPollingData"
    },
    "channelLifetime": "7200",
    "channelType": "LongPolling",
    "clientCorrelator": "123",
    "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123"
}}
```

D.3 Create Notification Channel (OMA Push method), using tel URI (section 6.1.5.2)

Request:

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: nnnn
Accept: application/json

{"notificationChannel": {
   "applicationTag": "myApp",
   "channelData": {
      “appId”: “x-wap-application:wml.ua”,
      "maxNotifications": "1",
      "type": "nc:OMAPushData"
   },
   "channelLifetime": "7200",
   "channelType": "OMAPush",
   "clientCorrelator": "987"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"notificationChannel": {
   "applicationTag": "myApp",
   "callbackURL": "http://example.com/callBackUrl/cbu222",
   "channelData": {
      “appId”: “x-wap-application:wml.ua”,
      "maxNotifications": "1",
      "type": "nc:OMAPushData"
   },
   "channelLifetime": "7200",
   "channelType": "OMAPush",
   "clientCorrelator": "987",
   "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch987"
}}
```

D.4 Create Notification Channel (Long Polling method), using ACR (section 6.1.5.3)

Request:

```
POST /exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels HTTP/1.1
Host: example.com:80
Content-Type: application/json
Content-Length: nnnn
```

```
Accept: application/json

{"notificationChannel": {
   "applicationTag": "myApp",
   "channelData": {
      "maxNotifications": "1",
      "type": "nc:LongPollingData"
   },
   "channelLifetime": "7200",
   "channelType": "LongPolling",
   "clientCorrelator": "123"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"notificationChannel": {
   "applicationTag": "myApp",
   "callbackURL": "http://example.com/callBackUrl/cbu111",
   "channelData": {
      "channelURL": " http://example.com/exampleAPI/notificationchannel/v1/ acr%3Apseudonym123/channels/ch123/notifications   ",
      "maxNotifications": "1",
      "type": "nc:LongPollingData"
   },
   "channelLifetime": "7200",
   "channelType": "LongPolling",
   "clientCorrelator": "123",
   "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch123"
}}
```

D.5 Create Notification Channel (OMA Push method), using ACR (section Error! Reference source not found.)

Request:

```
POST /exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels HTTP/1.1
Host: example.com:80
Content-Type: application/json
Content-Length: nnnn
Accept: application/json

{"notificationChannel": {
   "applicationTag": "myApp",
   "channelData": {
      “appId”: “x-wap-application:wml.ua”,
      "maxNotifications": "1",
      "type": "nc:OMAPushData"
   },
   "channelLifetime": "7200",
```

```
    "channelType": "OMAPush",
    "clientCorrelator": "987"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch987
Date: Thu, 04 Jun 2009 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"notificationChannel": {
    "applicationTag": "myApp",
    "callbackURL": "http://example.com/callBackUrl/cbu222",
    "channelData": {
        “appId”: “x-wap-application:wml.ua”,
        "maxNotifications": "1",
        "type": "nc:OMAPushData"
    },
    "channelLifetime": "7200",
    "channelType": "OMAPush",
    "clientCorrelator": "987",
    "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/acr%3Apseudonym123/channels/ch987"
}}
```

D.6 Retrieve individual Notification Channel (section 6.2.3.1)

Request:

```
GET /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

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

{"notificationChannel": {
    "applicationTag": "someOtherApp",
    "callbackURL": "http://example.com/callBackUrl/cbu333",
    "channelData": {
        "channelURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456/notifications",
        "maxNotifications": "5",
        "type": "nc:LongPollingData"
    },
    "channelLifetime": "7200",
    "channelType": "LongPolling",
    "clientCorrelator": "456",
    "resourceURL": "http://example.com/exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456"
```

```
}}
```

D.7 Removing Notification Channel (section 6.2.6.1)

Request:

```
DELETE /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch456 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2009 02:51:59 GMT
```

D.8 Single notification delivered including content (section 6.3.5.1)

Request:

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"longPollingRequestParameters": null}
```

Response:

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

{"notificationList": {"presenceNotification": {
   "callbackData": "1234",
   "link": {
      "href": "http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
tel%3A%2B19585550100/sub001",
      "rel": "PresenceSubscription"
   },
   "presence": {"person": {"mood": {"moodValue": "Happy"}}},
   "presentityUserId": "tel:+19585550100",
   "resourceStatus": "Active"
}}}
```

D.9 Multiple notifications delivered including content (section 6.3.5.2)

Request:

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"longPollingRequestParameters": null}
```

Response:

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

{"notificationList": [
  { "inboundMessageNotification": {"inboundMessage": {
      "destinationAddress": "tel:+19585550100",
      "inboundMMSMessage": {"subject": "Who is RESTing on the beach?"},
      "link": {
         "href": "http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123",
         "rel": "Subscription"
      },
      "messageId": "msg123",
      "resourceURL": "http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg123      ",
      "senderAddress": "tel:+19585550101"
   }}},
   {"inboundMessageNotification": {"inboundMessage": {
      "destinationAddress": "tel:+19585550100",
      "inboundMMSMessage": {"subject": "Who is still RESTing on the beach?"},
      "link": {
         "href": "http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123",
         "rel": "Subscription"
      },
      "messageId": "msg1234",
      "resourceURL": "http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg1234",
      "senderAddress": "tel:+19585550102"
   }}},
  { "presenceNotification": {
      "callbackData": "1234",
      "link": {
         "href": "http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
tel%3A%2B19585550100/sub001",
         "rel": "PresenceSubscription"
      },
      "presence": {"person": {"mood": {"moodValue": "Happy"}}},
      "presentityUserId": "tel:+19585550100",
      "resourceStatus": "Active"
   }
}]}
```

D.10 Server timeout (section 6.3.5.3)

Request:

```
POST /exampleAPI/notificationchannel/v1/tel%3A%2B19585550100/channels/ch123/notifications HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"longPollingRequestParameters": null}
```

Response:

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

{"notificationList": null}
```

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

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

Appendix F. Light-weight resources (Informative)

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

Appendix G. Authorization aspects (Normative)

This appendix specifies how to use the RESTful Notification Channel API in combination with some authorization frameworks.

G.1 Use with OMA Authorization Framework for Network APIs

The RESTful Notification Channel API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Notification Channel API supporting [Autho4API_10]:

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

G.1.1 Scope values

G.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 Notification Channel 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_notificationchannel.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_notificationchannel.longpoll	Provide access to all operations defined for using Long Polling on Notification Channel.	No
oma_rest_notificationchannel.omapush	Provide access to all operations defined for using OMA Push on Notification Channel.	No

Table 1: Scope values for RESTful Notification Channel API

G.1.1.2 Downscoping

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

- “oma_rest_notificationchannel.longpoll”
- “oma_rest_notificationchannel.omapush”

G.1.1.3 Mapping with resources and methods

Tables in this section specify how the scope values defined in section G.1.1.1for the RESTful Notification Channel API map to the REST resources and methods of this API. In these tables, the root "oma_rest_notificationchannel." of scope values is omitted for readability reasons.

Resource	URL Base URL: http://{serverRoot}/notificationchannel/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
Notification Channels	/{userId}/channels	6.1	**all_{apiVersion}** or **longpoll** or **omapush**	n/a	**all_{apiVersion}** or **longpoll** or **omapush**	n/a
Individual Notification Channel	/{userId}/channels/{channelId}	6.2	**all_{apiVersion}** or **longpoll** or **omapush**	n/a	n/a	**all_{apiVersion}** or **longpoll** or **omapush**

Table 2: Required scope values for: Management of Notification Channel

Resource	URL < specified by the server >	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
Notification list	<Resource URL is provided by the server when the Notification Channel is created>	6.3	n/a	n/a	**all_{apiVersion}** or **longpoll**	n/a

Table 3: Required scope values for: Retrieval of notifications from Notification Server

G.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 the{userId} resource URL variable in the resource URL path, when the RESTful Notification Channel API is used in combination with [Autho4API_10].

In the case the RESTful Notification Channel API supports [Autho4API_10], the server:

- SHALL accept ‘acr:auth’ as a valid value for the resource URL variable {userId}.

SHALL conform to [REST_Common_TS] section 5.8.1.1 regarding the processing of ‘acr:auth’

Appendix H. Notification server - Push enabler interaction (Informative)

This appendix provides further information on Notification Server Interaction with the Push Enabler for forwarding the event to the targeted device and application on the device.

In delivering the Push MESSAGE, the Notification Server has several implementation options:

a) Delivery via a Push Proxy Gateway (PPG) as defined in [OMA_PUSH], using either the Push Access Protocol [PushPAP] or the PushREST API [PushREST]. Depending upon the size of the notification and the intended bearer(s), the Notification Server may deliver the notification content directly, or provide an indirect reference to the notification content which the application may retrieve upon receiving the Push message. How the Notification Server determines the supported notification content size is unspecified, but as a general rule any notification content of less than 512 compressed/binary bytes or less than 2K uncompressed bytes should be deliverable via any OMA Push-OTA bearer binding.

Push PAP Example: Delivering Indirect Reference to Notification Content Available from Enabler Server

```
POST /pap HTTP/1.1
Content-Length: 1041
Content-Type: multipart/related; boundary=PMasdfglkjhqwert; type="application/xml"
Host: ppg.example.com:9002
Connection: close

--PMasdfglkjhqwert
Content-Type: application/xml

<?xml version="1.0"?>
<!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 1.0//EN" "http://www.wapforum.org/DTD/pap_1.0.dtd">
<pap product-name="OMA-Notification-Server-1.0">
<push-message push-id="1079025501l:mms_12.25.203.86_1223_1078969978_21:134:0:1"
 source-reference="notserver.example.com">
  <address address-value="WAPPUSH=+14255551212/TYPE=PLMN@example.com"/>
  <quality-of-service bearer="SMS" bearer-required="false" delivery-method="unconfirmed" network="GSM"
   network-required="false"/>
</push-message>
</pap>
--PMasdfglkjhqwert

Content-Length: 373
Content-Type: text/vnd.wap.si
X-Wap-Application-Id: myapp.com/f7adaea2-2bfe-1869-8314-1cc82b1aa4b8

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE si PUBLIC "-//WAPFORUM//DTD SI 1.0//EN" "http://www.wapforum.org/DTD/SI.dtd">
<si>
  <indication href="http://mmsapi.example.com/notification/myapp.com/f7adaea2-2bfe-1869-8314-1cc82b1aa4b8"
    si-id = "1079025501l:mms_12.25.203.86_1223_1078969978_21:134:0:1"
    >Your message was delivered.</indication>
</si>

--PMasdfglkjhqwert—
```

PushREST Example: Directly Delivering Notification Content

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com:9002
Content-Type: multipart/related; boundary=qwertyuioplkjhgfdsazxcvbnm; type="application/json"
Accept: application/json
Content-Length: 2794
Connection: close

--qwertyuioplkjhgfdsazxcvbnm
Content-Type: application/json

{"push-message": {
    "address": [
       {"address-value": "wappush=+14255551212/type=plmn@example.com "}
    ],
    "deliver-before-timestamp": "2010-11-08T18:13:51.0Z",
    "ppg-notify-requested-to": "http://notserver.example.com/Push/f7adaea2-2bfe-1869-8314-1cc82b1aa4b8",
    "progress-notes-requested": "true",
    "quality-of-service": {"priority": "medium", "bearer": "SMS" "bearer-required": "false" "delivery-method":
     "confirmed" "network": "GSM" "network-required": "false"},
    "source-reference": "notserver.example.com"
}}

--qwertyuioplkjhgfdsazxcvbnm
 Content-Type: application/xml
X-Wap-Application-Id: myapp.com/f7adaea2-2bfe-1869-8314-1cc82b1aa4b8

<?xml version="1.0" encoding="UTF-8"?>
<nc:notificationList xmlns:nc="urn:oma:xml:rest:netapi:notificationchannel:1">
  <pr:presenceNotification xmlns:pr="urn:oma:xml:rest:presence:1">
    <presentityUserId>tel:+19585550100</presentityUserId>
    <callbackData>1234</callbackData>
    <resourceStatus>Active</resourceStatus>
    <presence>
      <person>
        <mood>
          <moodValue>Happy</moodValue>
        </mood>
      </person>
    </presence>
    <link rel="PresenceSubscription"
href="http://example.com/exampleAPI/v1/presence/tel%3A%2B19585550101/subscriptions/presenceSubscriptions/
       tel%3A%2B19585550100/sub001"/>
</pr:presenceNotification>
<mms:inboundMessageNotification xmlns:mms="urn:oma:xml:rest:messaging:1">
    <inboundMessage>
      <destinationAddress>tel:+19585550100</destinationAddress>
      <senderAddress>tel:+19585550101</senderAddress>
      <resourceURL>http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg123
      </resourceURL>
      <link rel="Subscription" href="http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123"/>
      <messageId>msg123</messageId>
      <inboundMMSMessage>
        <subject>Who is RESTing on the beach?</subject>
      </inboundMMSMessage>
```



```
   </inboundMessage>
</mms:inboundMessageNotification>
 <mms:inboundMessageNotification xmlns:mms="urn:oma:xml:rest:messaging:1">
   <inboundMessage>
    <destinationAddress>tel:+19585550100</destinationAddress>
    <senderAddress>tel:+19585550102</senderAddress>
    <resourceURL>http://example.com/exampleAPI/v1/messaging/inbound/registrations/reg123/messages/msg1234
    </resourceURL>
    <link rel="Subscription" href="http://example.com/exampleAPI/v1/messaging/inbound/subscriptions/sub123"/>
    <messageId>msg1234</messageId>
    <inboundMMSMessage>
     <subject>Who is still RESTing on the beach?</subject>
    </inboundMMSMessage>
   </inboundMessage>
</mms:inboundMessageNotification>
</nc:notificationList>
--qwertyuioplkjhgfdsazxcvbnm--
```

b) Direct delivery of an OMA Push message using a Push-OTA (Over the Air) binding supported by the target device. How the Notification Server determines the supported Push-OTA bindings is unspecified. For details of Push-OTA bearer bindings, see [PushOTA].