RESTful Network API for OMA Push

Approved Version 1.0 – 29 Oct 2013

Open Mobile Alliance

OMA-TS-REST_NetAPI_Push-V1_0-20131029-A

Contents

Figures

Tables

1. Scope

This specification defines a RESTful API using HTTP protocol bindings, based upon the OMA Push Access Protocol (PAP). The RESTful Network API for Push is based upon PAP as defined in the OMA Push [Push2.3] enabler release.

2. References

2.1 Normative References

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

[PPGService] "Push Proxy Gateway Service Specification". Open Mobile Alliance™. OMA-TS-PPGService-V2_3. URL:http://www.openmobilealliance.org/

[REST_TS_Common] "Common definitions for OMA RESTful Network APIs", Open Mobile Alliance™, OMA-TS-REST_NetAPI_Common-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

[RFC2387] "The MIME Multipart/Related Content-type", E. Levinson, August 1998, URL: http://www.ietf.org/rfc/rfc2387.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee et al. August 1998. URL: http://www.ietf.org/rfc/rfc2396.txt

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

[SUP_Common] "XML schema for Common definitions for RESTful Network APIs". Open Mobile Alliance™. OMA-SUP-XSD_rest_netapi_common-V1_0. URL: http://www.openmobilealliance.org/

[SUP_NetAPI_Push] "XML schema for the RESTful Network API for OMA Push - Push Access Protocol". Open Mobile Alliance™. OMA-SUP-XSD_rest_push-V1_0. URL: http://www.openmobilealliance.org/

[WDP] "Wireless Datagram Protocol". WAP Forum™, WAP-259-WDP. URL: http://www.openmobilealliance.org/

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

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

2.2 Informative References

[EMN] "Email Notification" Version 1.0. OMA-Push-EMN-V1_0, Open Mobile Alliance™. URL: http://www.openmobilealliance.org/

[EMNEnabler] "Enabler Release Definition Email Notification" Version 1.0. OMA-ERELD-EMN-V1_0, Open Mobile Alliance™. URL: http://www.openmobilealliance.org/

[OMADICT] "Dictionary for OMA Specifications", Version 2.8, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_8, URL:http://www.openmobilealliance.org/

[OMNA] "OMA Naming Authority". Open Mobile Alliance™. URL: http://www.openmobilealliance.org/OMNA.aspx

[Push2.3] "Enabler Release Definition for Push Version 2.3", Open Mobile Alliance™. OMA-ERELD-Push-V2_3. URL: http://www.openmobilealliance.org/

[PushArch] "Push Architectural Overview". Open Mobile Alliance™. OMA-AD-Push-V2_3
URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

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

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

3.2 Definitions

Application	Use definition from [PAP].
Base URI	An HTTP URI as described in [RFC2616] which specifies REST resources relating to a service of the RESTful Network API for OMA Push.
Client	Use definition from [PAP].
Domain	Use definition from [OMADICT].
Enabler	Use definition from [OMADICT].
Push Initiator	Use definition from [PAP].
Push Proxy Gateway	Use definition from [PAP].
Resource	Use definition from [RFC2616].
QoS (Quality of Service)	Use definition from [OMADICT].
Uniform Resource Identifier	Use definition from [OMADICT].

3.3 Abbreviations

API	Application Programming Interface
AS	Application Server
CDMA	Code Division Multiple Access
GSM	Global System for Mobile Telecommunication
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
PAP	Push Access Protocol
PI	Push Initiator
PPG	Push Proxy Gateway
REST	REpresentational State Transfer
SCR	Static Conformance Requirements
SIP	Session Initiation Protocol
SMS	Short Message Service
TS	Technical Specification
UE	User Equipment

UMTS	Universal system for Mobile Telecommunication
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WWAN	Wireless Wide Area Network
WP	White Paper
XML	eXtensible Markup Language
XSD	XML Schema Definition

4. Introduction

The Technical Specification of the RESTful Network API for OMA Push contains HTTP protocol bindings for the OMA Push Access Protocol [PAP], using the REST architectural style. The binding has been designed in a way that maximizes PAP re-use, at the same time applying RESTful principles as much as possible under the premise of re-use.

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. The goal of the RESTful Network API for OMA Push is to provide a simple, easy to use, uniform interface between a Push Initiator, and Push Proxy Gateway; whether the PI is realized on a UE, or an AS.

4.1 Version 1.0

Version 1.0 of this specification supports the following operations.

The Push Initiator (PI) is able to initiate the following operations to the Push Proxy Gateway (PPG):

- Push Submission
- Push Submission with Replace
- Push Cancellation
- Status Query
- Client Capabilities Query

The PPG is able to initiate the following message to the PI:

- Result Notification

5. Push API definition

This section is organized to support a comprehensive understanding of the Push API design. It specifies the entire domain of the Push Network API, including:

- all resources
- data structures
- methods permitted on the specified resources.

The following aspects are based upon the definitions in [REST_TS_Common]:

- Error handling: per “5.2 Unsupported Formats”
- Naming conventions: per “5.3 Authoring Style”, with the exception that the names of XML elements and attributes originating from [PAP] are used in their original spelling
- Content type negotiation: per section “5.4 Content type negotiation” in case both JSON and XML are supported, however the statement “e” regarding the format of notifications does not apply.
- Resource creation: per “5.5 Resource creation”
- JSON message body formatting: per “5.6 JSON encoding in HTTP Requests/Responses”

The remainder of this document is structured as follows:

Section 5 begins with a table listing all PushREST API resources, the corresponding resource URIs and data structures Supported HTTP verbs are as defined in section 5.1 “Resources Summary”. In addition, for each supported PushREST resource/verb combination, the table lists the equivalent PAP operation (i.e. where applicable). What follows are the data structures (section 5.2). A sample of typical use cases utilizing flow diagrams are defined in section 5.3 “Sequence Diagrams”, described as high level flow diagrams.

Section 6 contains the detailed specification for each of the resources. Each subsection defines the resource, the request URI 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. JSON examples are provided in Appendix C. Appendix B provides the Static Conformance Requirements (SCR).

Note that the Push Network API re-uses the data structures from the Push Access Protocol [PAP] as far as possible, and provides a mapping from the PAP operations to the RESTful style which minimizes the need for additional storage of status information. Note further that the “push-id” attribute from PAP is included as “pushId” in the resource URL, rather than in the data structures.

Note further that PAP wraps every data structure with a “<pap>” XML element that is not used in the Push Network API, because it is seen easier for the developers to directly access the data structures representing the individual REST resources. As a consequence of omitting the wrapper, also the “product-name” attribute from PAP is not supported in the Push Network API. This attribute was introduced in the early stages of PAP to fix interoperability glitches between implementations; however, it is no longer needed in today’s mature standardized environment.

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).

For requests and responses that have a body, the following applies: the PPG SHALL support XML and MAY support JSON as encoding formats of the parameters in the body. In case the PPG supports both formats:

- In the response body, it 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_TS_Common].
- In notifications to the PI, the PPG SHALL use either XML or JSON encoding, depending on which format the PI has used in the related request that created the subscription.

5.1 Resources Summary

This section summarizes all the resources used by the Push 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 illustrates 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.

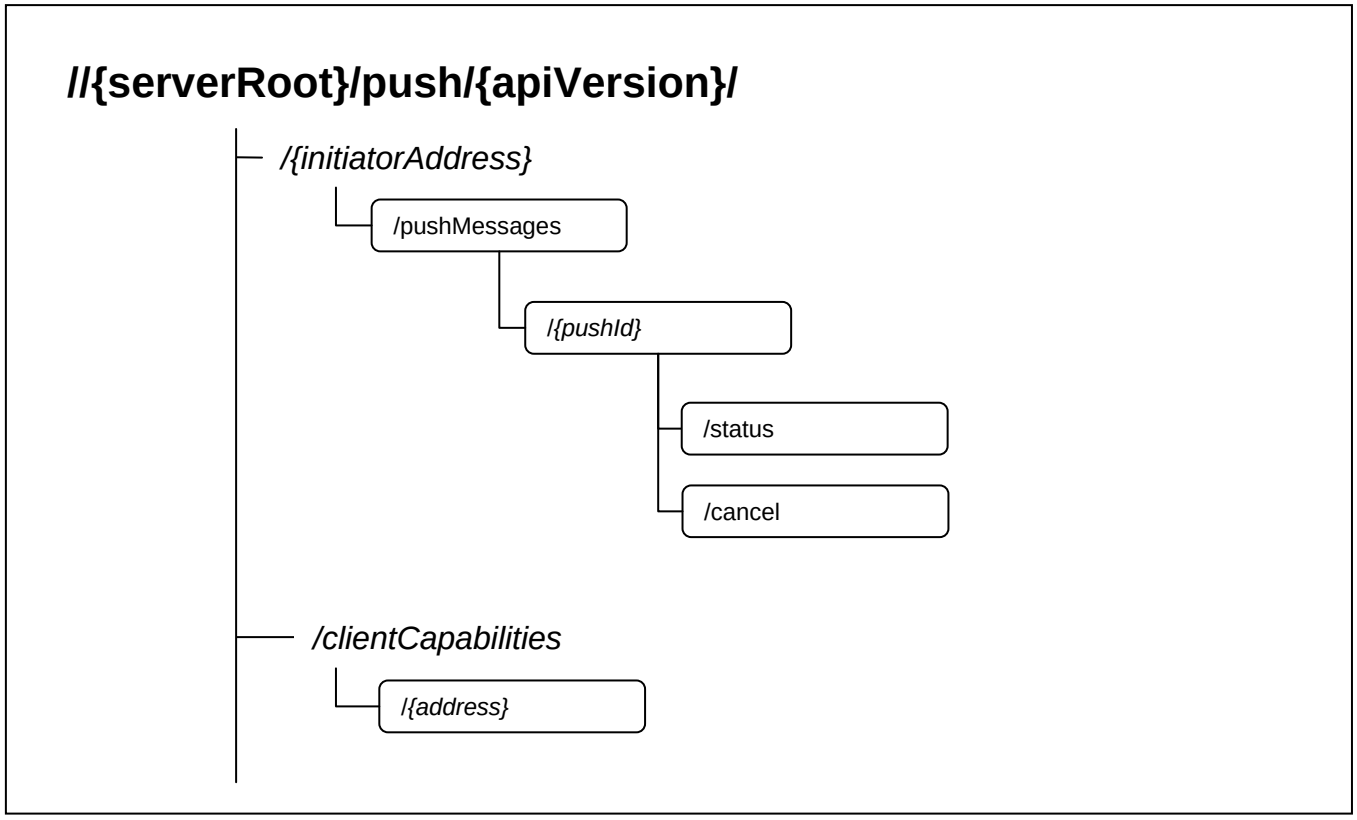

Figure 1: Resource structure defined by this specification

5.1.1 Base URI

The Push Network API SHALL utilize the following Base URI – "http://{serverRoot}/push/{apiVersion}" whereby:

- serverRoot – a part of an HTTP URI as specified in [RFC2616] Section "*Uniform Resource Identifiers*" which identifies a Push API service (an authority – (host and port) plus optional base path), as assigned by the Push API service provider; and
- apiVersion – a constant string which specifies a Push API version, in this version set to "v1", which supports the features of the PAP interface as of [Push2.3].

Note: adding support for the Push Network API does neither impact nor require support for the PAP interface (either Push 2.3 or earlier versions) by either PI or PPG. The Push Network API is a new, supplemental method of accessing the same services available in Push 2.3 or earlier versions of the OMA Push enabler.

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. The "PAP" row indicates the PAP equivalent operation.

Purpose: Sending Push Message, replacing Push Message, obtaining the delivery status, and cancelling the message

Resource	URL Base URL: http://{serverRoot}/push/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Push messages	/{initiatorAddress}/pushMessages/{pushId}	push-message (used in MIME multipart body for PUT request) push-response (used in PUT response) cancel-response (used DELETE response)	no	Create new Push message, or replace a Push message	no	Cancel the whole Push message
				PAP: push-message		PAP: cancel-message
Push message status	/{initiatorAddress}/pushMessages/{pushId}/status	statusquery-response (used in GET response)	Query the status of the Push message	no	no	no
			PAP: statusquery-message			
Partial Push message cancellation	/{initiatorAddress}/pushMessages/{pushId}/cancel	cancel-message (used in POST request) cancel-response (used in POST response)	no	no	Cancel the Push message for some addresses	no
					PAP: cancel-message	

Table 1 Overview of Resources for Creating, Status Query, and Cancellation of Push Messages

Purpose: Callback notifications for Push Message delivery status

Resource	URL <specified by the PI>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
PI notification about Push message delivery status	<specified by the PI when Push message is created or updated>	resultnotification-response	no	no	Notify PI of Push message result	no
					PAP: resultnotification-response	

Table 2 Overview of Resources for Delivery Status Notifications

Purpose: Query client capabilities

Resource	URL Base URL: http://{serverRoot}//push{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Client capabilities	/clientCapabilities/{address}	ccq-response (used in GET response)	Query client capabilities	no	no	no
			PAP: ccq-message			

Table 3 Overview of Resources for Client Capability Query

5.2 Data Types

5.2.1 XML Namespaces

The namespace for the data types in the Push Network API is:

urn:oma:xml:rest:netapi:push:1

The 'xsd' namespace is used in the present document to refer to the XML Schema data types defined in XML Schema [XMLSchema1, XMLSchema2]. The 'common' namespace is used in the present document to refer to the data types defined in [REST_TS_Common] and [SUP_Common]. The use of the names 'xsd' and ‘common’ is not semantically significant.

The XML schema for the data structures defined in the section below are given in [SUP_NetAPI_Push].

5.2.2 Structures

The subsections of this section define the data structures used in the Push Network API.

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

5.2.2.1 Type: push-message-type

This data type represents the control entity of a Push message. It is inherited from [PAP].

Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- “push-id”, as it is part of the resourceURL

Further note that replace-push-id from PAP maps in the Push Network API to a URL pointing to the resource representing the original push message.

Element/Attribute	Type	XML	Optional	Description
address	address-type [1..unbounded]	E	No	Target address(es).
quality-of-service	quality-of-service-type	E	Yes	Delivery qualities desired by the PI.
replace-push-message	xsd:anyURI	A	Yes	This attribute contains a URL pointing to the resource which represents the Push message to be replaced. It has the same purpose as the replace-push-id attribute defined in [PAP].
replace-method	xsd:string	A	Yes	See [PAP].
deliver-before-timestamp	xsd:datetime	A	Yes	See [PAP].
deliver-after-timestamp	xsd:datetime	A	Yes	See [PAP].
source-reference	xsd:string	A	Yes	See [PAP].
ppg-notify-requested-to	xsd:anyURI	A	Yes	See [PAP]. The entity body format of resultnotification messages (see section 5.2.2.12) related to this push message SHALL be the same as the entity body format of the related push-response message, i.e. either application/xml or application/json.
progress-notes-	xsd:boolean	A	Yes	See [PAP].

requested				

Table 4 Type: push-message-type

The format of the Push message entity in a PUT request is a MIME multipart/related [RFC2387] compound object that includes the control entity which is an XML element named “push-message” of type “push-message-type”, the content to be pushed, and an OPTIONAL capabilities entity [PAP].

5.2.2.2 Type: push-response-type

This data type represents the response of creating or updating a Push message. It is inherited from [PAP].

Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- “push-id”, as it is part of the resourceURL

Element/Attribute	Type	XML	Optional	Description
progress-note	progress-note-type [0..unbounded]	E	Yes	See [PAP].
response-result	response-result-type	E	No	See [PAP].
resourceURL	xsd:anyURI	E	No	Link to the resource representing the Push message.
sender-address	xsd:string	A	Yes	This OPTIONAL field contains the same data as the resourceURL and is included here for backwards-compatibility reasons. Implementations MAY therefore omit it. See also [PAP].
sender-name	xsd:string	A	Yes	See [PAP].
reply-time	xsd:datetime	A	Yes	See [PAP].

Table 5 Type: push-response-type

An XML root element named “push-response” of type “push-response-type” is allowed in response bodies.

5.2.2.3 Type: address-type

This data type represents an address. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
address-value	xsd:string	A	No	The target device address for use by the PPG, formatted per [PPGService].

Table 6 Type: address-type

5.2.2.4 Type: quality-of-service-type

This data type represents a set of quality-of-service parameters. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
priority	priority-type	A	Yes	See [PAP].
delivery-method	delivery-method-type	A	Yes	See [PAP].
network	network-type	A	Yes	Adapted from [PAP]. The value MAY be one of the values defined by the network-type enumeration, or unspecified values for other network types as possibly supported in the future. Omitting this attribute in a request from the PI to the PPG means that the PPG can freely choose the network.
network-required	xsd:boolean	A	Yes	See [PAP].
bearer	bearer-type	A	Yes	Adapted from [PAP]. The value MAY be one of the values defined by the bearer-type enumeration, or unspecified values for other bearer types as possibly supported in the future. Omitting this attribute in a request from the PI to the PPG means that the PPG can freely choose the bearer.
bearer-required	xsd:boolean	A	Yes	See [PAP].

Table 7 Type: quality-of-service-type

5.2.2.5 Type: progress-note-type

This type defines a progress note parameters. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
stage	xsd:string	A	No	See [PAP].
note	xsd:string	A	Yes	See [PAP].
time	xsd:dateTime	A	Yes	See [PAP].

Table 8 Type: progress-note-type

5.2.2.6 Type: response-result-type

This type defines a response result. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 9 Type: response-result-type

5.2.2.7 Type: cancel-message-type

This data type represents the parameters to be submitted to cancel a Push message. It is inherited from [PAP].
Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- "push-id", as it is part of the resourceURL.

Element/Attribute	Type	XML	Optional	Description
address	address-type [0..unbounded]	E	Yes	See [PAP]. Omitting this element implies cancelling the Push message for all addresses.

Table 10 Type: cancel-message-type

An XML root element named "cancel-message" of type "cancel-message-type" is allowed in HTTP request entity bodies.

5.2.2.8 Type: cancel-response-type

This data type represents the response of cancelling a Push message. It is inherited from [PAP].
Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- "push-id", as it is part of the resourceURL

Element/Attribute	Type	XML	Optional	Description
cancel-result	cancel-result-type [1..unbounded]	E	No	See [PAP].
resourceURL	xsd:anyURI	E	No	Link to the resource representing the Push message that has been cancelled.

Table 11 Type: cancel-response-type

An XML root element named "cancel-response" of type "cancel-response-type" is allowed in response bodies.

5.2.2.9 Type: cancel-result-type

This data type represents the result of cancelling a Push message for a set of addresses. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
address	address-type [0..unbounded]	E	Yes	See [PAP]. Omitting this element implies all addresses of this particular Push message.
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 12 Type: cancel-result-type

5.2.2.10 Type: statusquery-response-type

This data type represents the current status of a Push message. It is inherited from [PAP].

Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- "push-id", as it is part of the resourceURL

Element/Attribute	Type	XML	Optional	Description
statusquery-result	statusquery-result-type [1..unbounded]	E	No	See [PAP].
resourceURL	xsd:anyURI	E	No	Link to the resource representing the status of the Push message.

Table 13 Type: statusquery-response-type

An XML root element named "statusquery-response" of type "statusquery-response-type" is allowed in response bodies.

5.2.2.11 Type: statusquery-result-type

This data type represents the current status of a Push message for a set of addresses. Its structure is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
address	address-type [0..unbounded]	E	Yes	See [PAP]. Omitting this element implies all addresses of this particular Push message.
quality-of-service	quality-of-service-type	E	Yes	See [PAP].
event-time	xsd:dateTime	A	Yes	See [PAP].
message-state	State	A	No	See [PAP].
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 14 Type: statusquery-result-type

5.2.2.12 Type: resultnotification-message-type

This data type is used in notifications from the PPG to the PI to specify the outcome of a submitted message for a specific recipient after the final result is known. It is inherited from [PAP].

Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- "push-id", as it is part of the resource URL in the link element.

Element/Attribute	Type	XML	Optional	Description
address	address-type	E	No	See [PAP].
link	common:Link	E	No	Link to the Push message resource to which the notification relates. See [REST_TS_Common]. Valid values for the "rel" attribute are defined in section 5.2.4.

quality-of-service	quality-of-service-type	E	Yes	See [PAP].
successful-recipients	address-type	E	Yes	See [PAP]. Introduced in PAP 2.3.
unsuccessful-recipients	address-type	E	Yes	See [PAP]. Introduced in PAP 2.3.
sender-address	xsd:string	A	Yes	This OPTIONAL field contains the same data as the href attribute of the link element and is included here for backwards-compatibility reasons. Implementations MAY therefore omit it. See also [PAP].
sender-name	xsd:string	A	Yes	See [PAP].
received-time	xsd:dateTime	A	Yes	See [PAP].
event-time	xsd:dateTime	A	Yes	See [PAP].
message-state	State	A	No	See [PAP].
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 15 Type: resultnotification-message-type

An XML root element named “resultnotification-message” of type “resultnotification-message-type” is allowed in notification HTTP request entity bodies.

A PI needs to be prepared that not all PPGs support the elements introduced in PAP 2.3, and therefore SHOULD NOT rely on them.

If the “delivery-method” attribute was set to "confirmed-with-response" in the corresponding Push message's “quality-of-service” element, and the PPG received content from the terminal, the “resultnotification-message” and the content are sent together in a multipart/related entity body. Otherwise, the “resultnotification-message” is sent as a plain application/xml entity body.

5.2.2.13 Type: resultnotification-response-type

This data type is used in responses sent by the PI to the PPG to acknowledge a notification. It is inherited from [PAP].

Note that the following items from [PAP] are not instantiated in the Push Network API:

- “push-id” attribute, needed in PAP (together with “address” below) to correlate the resultnotification-response to the corresponding resultnotification-message
- “address” element, needed in PAP (together with “push-id” above) to correlate the resultnotification-response to the corresponding resultnotification-message

Correlating a resultnotification-response to the corresponding resultnotification-message is not needed in the Push Network API, because the resultnotification-response structure is returned immediately in the HTTP response to the resultnotification-message.

Element/Attribute	Type	XML	Optional	Description
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 16 Type: resultnotification-response-type

An XML root element named “resultnotification-response” of type “resultnotification-response-type” is allowed in notification HTTP response entity bodies.

5.2.2.14 Type: ccq-response-type

This data type is used in response messages from the PPG to the PI to a client capabilities query for a specified device. It is inherited from [PAP].

Note that the following attributes from [PAP] are not instantiated in the Push Network API:

- “query-id”, because it is only used for correlating request and response which is not needed in the Push Network API.

Element/Attribute	Type	XML	Optional	Description
address	address-type	E	No	See [PAP].
resourceURL	xsd:anyURI	E	No	Link to the resource representing the client’s address.
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].

Table 17 Type: ccq-response-type

An XML root element named “ccq-response” of type “ccq-response-type” is allowed in response bodies

5.2.2.15 Type: badmessage-response-type

This data type is used in the response sent by the PPG to PI to notify that the messages are unrecognisable or that are of a protocol version that is not supported. It is inherited from [PAP].

Element/Attribute	Type	XML	Optional	Description
code	xsd:string	A	No	See [PAP].
desc	xsd:string	A	Yes	See [PAP].
bad-message-fragment	xsd:string	A	Yes	See [PAP].

Table 18 Type: badmessage-response-type

An XML root element named “badmessage-response” of type “badmessage-response-type” is allowed in response bodies.

5.2.3 Enumerations

The subsections of this section define the enumerations used in the Push Network API.

5.2.3.1 Enumeration: priority-type

This enumeration defines priority values. Its content is inherited from [PAP].

Enumeration	Description
High	See [PAP].
Medium	See [PAP].
Low	See [PAP].

Table 19 Enumeration: priority-type

5.2.3.2 Enumeration: delivery-method-type

This enumeration defines delivery types. Its content is inherited from [PAP].

Enumeration	Description
Confirmed	See [PAP].
preferconfirmed	See [PAP].
confirmed-with-response	See [PAP]. Introduced in PAP 2.1.
oneshot	See [PAP]. Introduced in PAP 2.1.
unconfirmed	See [PAP].
notspecified	See [PAP].

Table 20 Enumeration: delivery-method-type

A PI needs to be prepared that not all PPGs support the values introduced in PAP 2.1, and can respond with an error message in that case.

5.2.3.3 Enumeration: replace-method-type

This enumeration defines replace methods. Its content is inherited from [PAP].

Enumeration	Description
pending-only	See [PAP].
all	See [PAP].

Table 21 Enumeration: replace-method-type

5.2.3.4 Enumeration: network-type

This enumeration defines network types. Some basic values are inherited from [WDP] which is referenced from [PAP]. However, as the list of network values in [WDP] is outdated, this specification provides a simplified set that covers today's market realities. Further values from [WDP] MAY be supported by implementations.

Enumeration	Description
WWAN	Wireless Wide Area Network, subsuming multiple technologies such as e.g. GSM, CDMA, UMTS, LTE, WiMAX. The actual network technology is chosen by the PPG.
GSM	Legacy value from [WDP] to represent GSM networks.
IS-95 CDMA	Legacy value from [WDP] to represent CDMA networks.
WLAN	Wireless Local Area Network

Table 22 Type: network-type

5.2.3.5 Enumeration: bearer-type

This enumeration defines bearer types. Some basic values are inherited from [WDP] which is referenced from [PAP]. However, as the list of bearer values in [WDP] is outdated, this specification provides a simplified set that covers today's market realities. Further values from [WDP] MAY be supported by implementations.

Enumeration	Description
SMS	Push over SMS (from [WDP])
CBS	Push over Cell Broadcast
IP	Push over Internet Protocol
SIP	Push over SIP
MBMS	Multipoint Push over MBMS
BCAST	Multipoint Push over OMA BCAST

Table 23 Type: bearer-type

5.2.4 Values of the Link "rel" attribute

The "rel" attribute of the link element (see [REST_TS_Common]) 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, indicating resources that are defined in this specification which the "link" element can point to (list is non-exhaustive, and can be extended):

- push-message

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

5.3 Sequence Diagrams

This section summarizes various sequence flows for various use cases of the Push Network API.

5.3.1 Push message submission and status query

The figure below shows a scenario for submitting a push message and check for delivery status.

The used resources are:

- To submit a push message, create a new resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

- To get the delivery status of the message, read the resource

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}/status

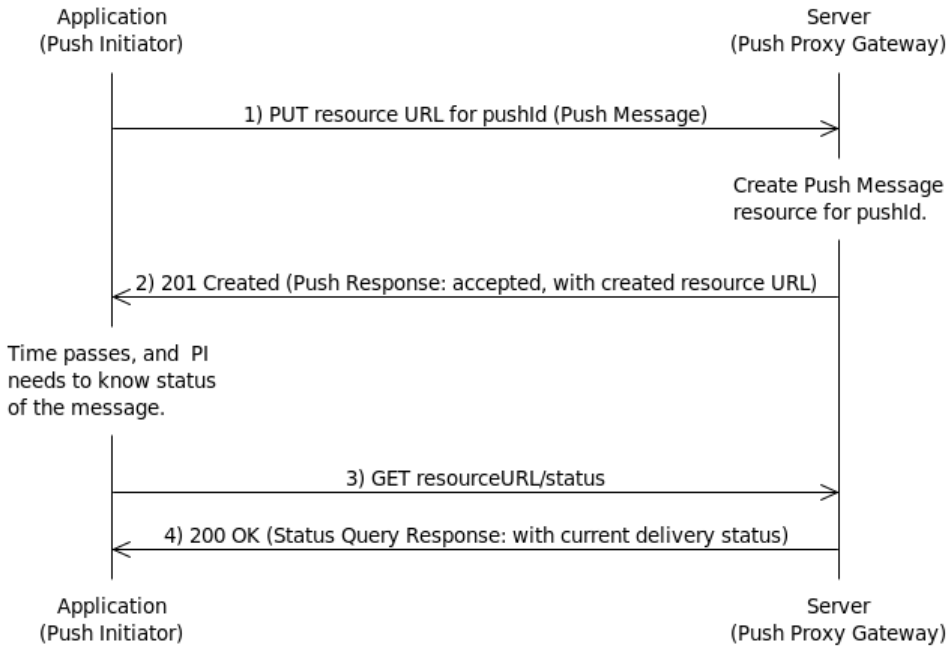

Figure 2: Push message submission and status check

1. As Push Initiator, an Application requests Push message delivery using PUT for a new resource identified by the initiatorAddress and pushId.
2. The PPG creates the new resource, and confirms Push message acceptance for the pushId.
3. The application requests the delivery status of the Push message created earlier, using GET to the resource URL created earlier.
4. The PPG responds with the delivery status for the Push message.

5.3.2 Push message submission and result notification

The figure below shows a scenario for submitting a push message with request for result notification, and later receiving the notification.

The used resources are:

- To submit a push message, create a new resource using the PUT method

http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

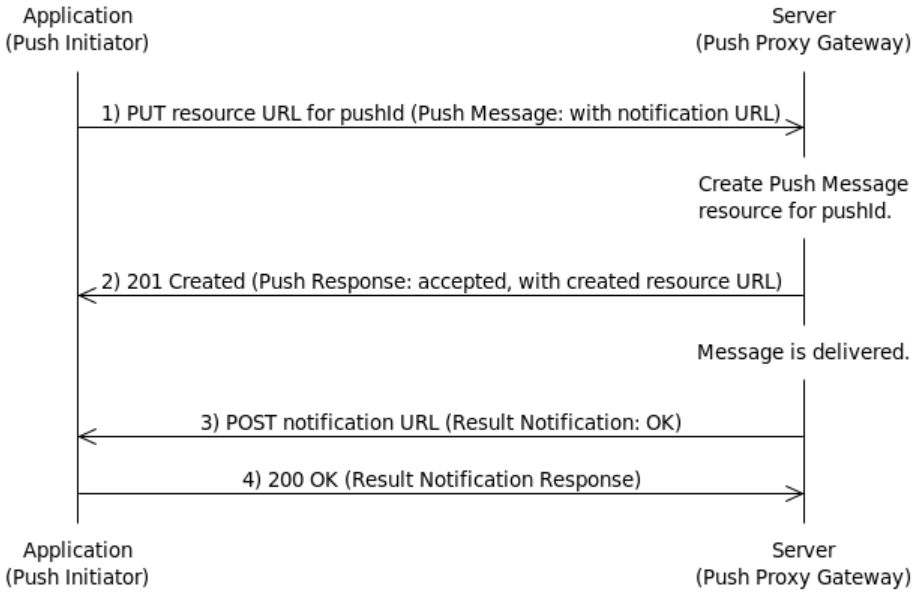

Figure 3: Push message submission and result notification

1. As Push Initiator, an application requests Push message delivery using PUT for a new resource URL uniquely identified by the initiatorAddress and pushId, and includes a result notification URL indicating that the application wants to be explicitly informed of the Push Message delivery result.
2. The PPG creates the new resource, and confirms Push Message acceptance for the pushId.
3. The PPG completes delivery of the Push message, and sends a result notification to the notification URL indicating that the message was successfully delivered, and further details as applicable (e.g. to which target addresses the delivery was successful).
4. The PI responds with a result notification response confirming receipt of the notification.

5.3.3 Push message submission and replacement using same pushId

The figure below shows a scenario for submitting a push message, and later replacing the push message with another message using the same pushId, prior to delivery completion of the original message.

The used resources are:

- To submit a push message, create a new resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

- To replace a push message using the same pushId, replace the earlier created resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

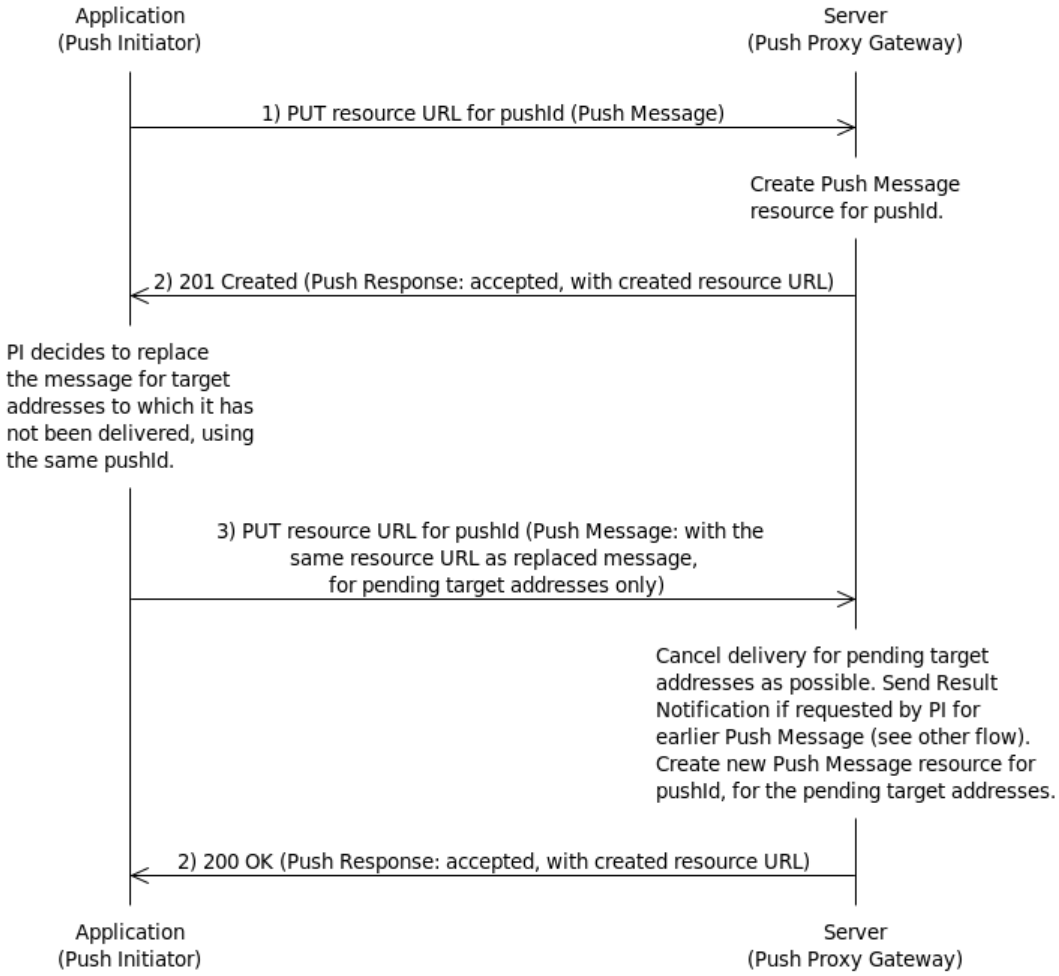

Figure 4: Push message submission and replacement using same pushId

1. As Push Initiator, an application requests Push message delivery using PUT for a new resource URL uniquely identified by the initiatorAddress and pushId.
2. The PPG creates the new resource, and confirms Push Message acceptance for the pushId.
3. The PI decides to replace the earlier Push Message for all recipients for which delivery has not been completed, and requests Push message replacement using PUT for the same resource URL earlier created for the pushId, and identifying the earlier Push Message resource URL as the message to be replaced.
4. The PPG cancels delivery of the earlier Push Message for all pending target addresses, if possible, and sends a Result Notification if requested by the PI for the earlier Push Message (see section 5.3.2). The PPG replaces the Push Message resource for the pushId with a new Push Message for the pending target addresses only, and confirms acceptance of Push Message replacement for the pushId.

5.3.4 Push message submission and replacement using new pushId

The figure below shows a scenario for submitting a push message, and later replacing the push message with another message with a different pushId, prior to delivery completion of the original message.

The used resources are:

- To submit a Push Message, create a new resource using the PUT method

http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId1}

- To replace a Push Message with a new Push Message using a new pushId, create a new resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId2}

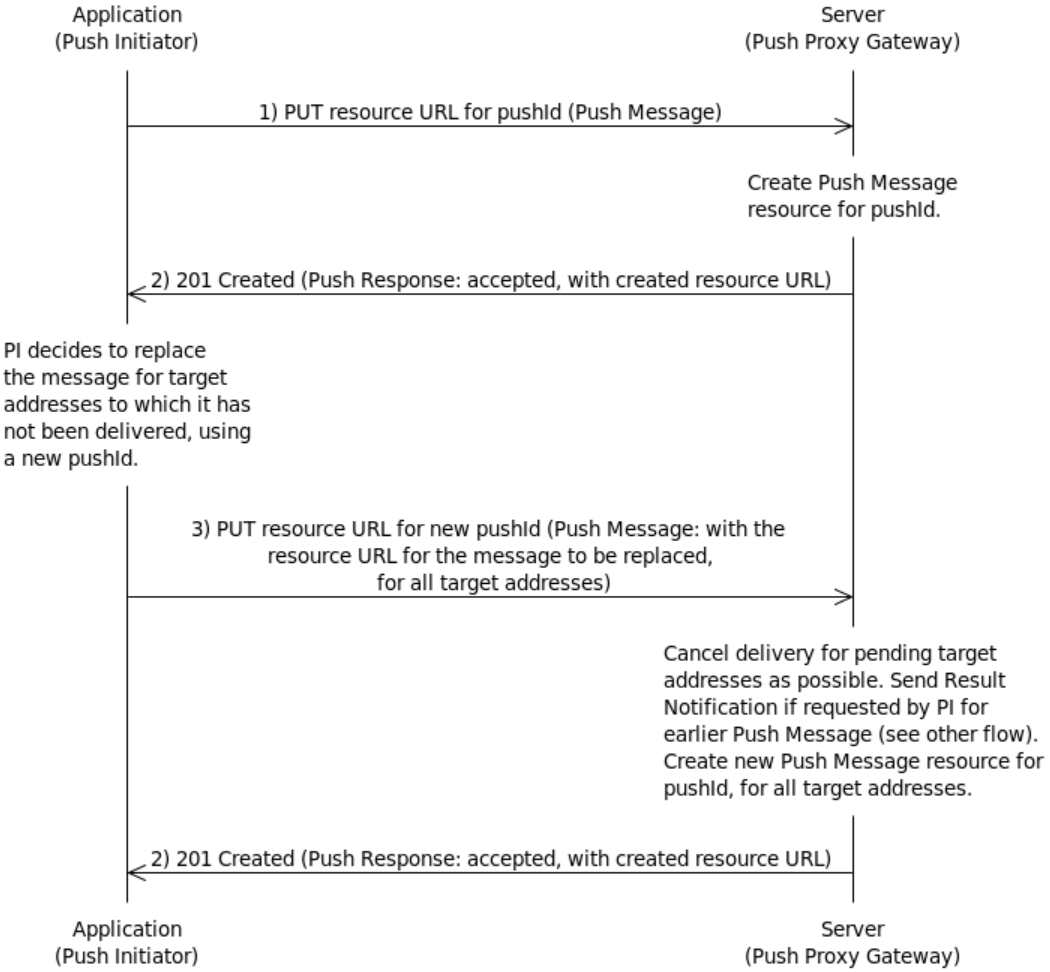

Figure 5: Push message submission and replacement using new pushId

1. As Push Initiator, an application requests Push message delivery using PUT for a new resource URL uniquely identified by the initiatorAddress and pushId.
2. The PPG creates the new resource, and confirms Push Message acceptance for the pushId.
3. The PI decides to replace the earlier Push Message for all recipients, and requests Push message replacement using PUT for a new resource URL uniquely identified by the initiatorAddress and the new pushId. The body of the request contains reference that identifies the earlier Push Message resource URL as the message to be replaced.
4. The PPG cancels delivery of the earlier Push Message for all pending target addresses, if possible, and sends a Result Notification if requested by the PI for the earlier Push Message (see section 5.3.2). The PPG creates a new Push Message resource for the new pushId, for all of the target addresses, and confirms acceptance of Push Message replacement for the new pushId.

5.3.5 Push message submission and cancellation

The figure below shows a scenario for submitting a push message and later cancelling it.

The used resources are:

- To submit a push message, create a new resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

- To cancel the message, delete the resource using the DELETE method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

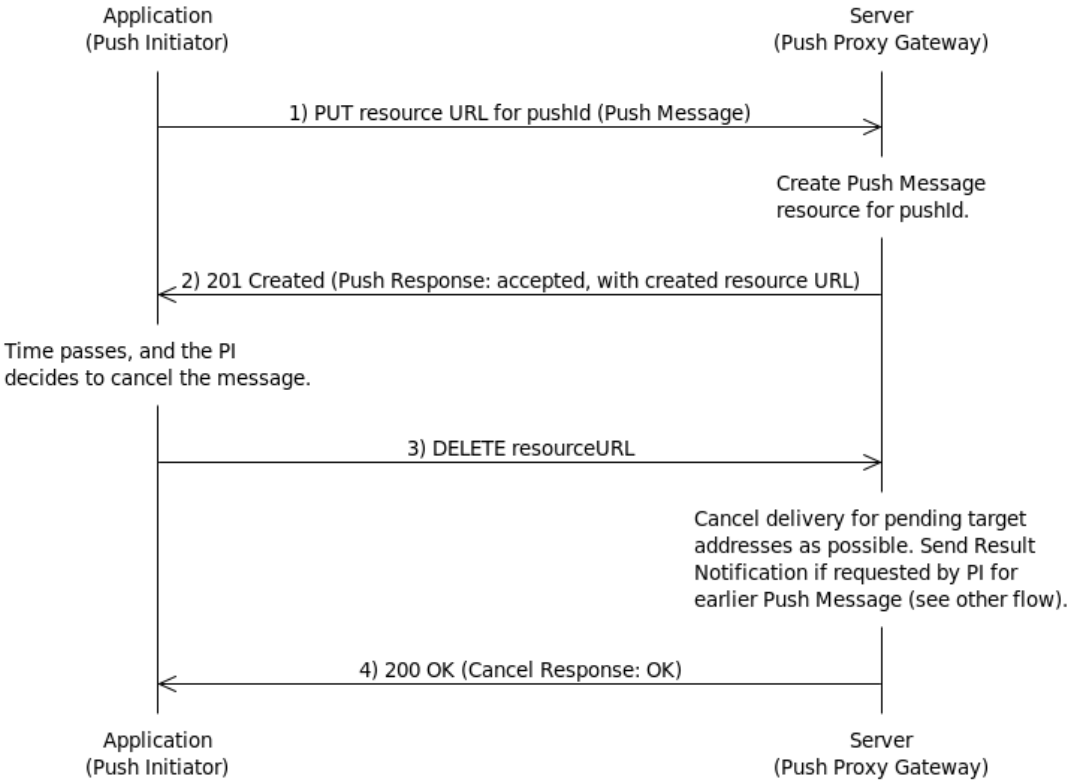

Figure 6: Push message submission and cancellation

1. As Push Initiator, an application requests Push message delivery using PUT for a new resource URL uniquely identified by the initiatorAddress and pushId.
2. The PPG creates the new resource, and confirms Push Message acceptance for the pushId.
3. After some time, the application decides to cancel the Push Message created earlier, using DELETE to the resource URL created earlier.
4. The PPG cancels delivery of the earlier Push Message if possible for all pending target addresses and sends a Result Notification if requested by the PI for the earlier Push Message (see section 5.3.2). The PPG responds with confirmation of the Push Message cancellation.

5.3.6 Push message submission and partial cancellation

The figure below shows a scenario for submitting a push message and later cancelling it for some target addresses.

The used resources are:

- To submit a push message, create a new resource using the PUT method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

- To cancel the message for some target addresses, send a Cancel Message request using the POST method

  http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}/cancel

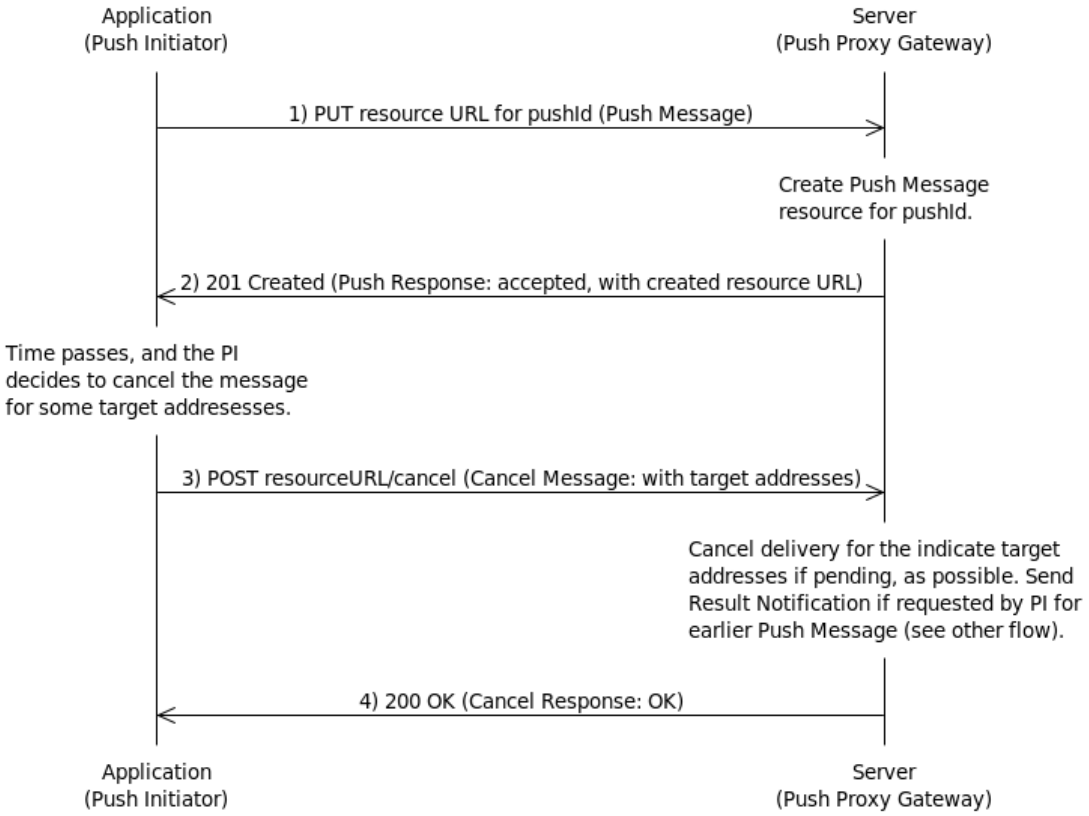

Figure 7: Push message submission and partial cancellation

1. As Push Initiator, an application requests Push message delivery using PUT for a new resource URL uniquely identified by the initiatorAddress and pushId.
2. The PPG creates the new resource, and confirms Push Message acceptance for the pushId.
3. After some time, the application decides to cancel the Push Message created earlier for some target addresses only. The application sends a Cancel Message request using POST to the resource URL created earlier, indicating the set of addresses for which message delivery should be cancelled.
4. The PPG cancels delivery of the earlier Push Message if possible for all of the indicated target addresses for which delivery is still pending, and sends a Result Notification if requested by the PI for the earlier Push Message (see section 5.3.2). The PPG responds with confirmation of the Push Message cancellation for the indicated target addresses.

5.3.7 Query client capabilities

This figure below shows a scenario for querying the Push client capabilities information for a target address.

The used resources are:

- To obtain the client capabilities, send a Client Capability Query request to the resource URL.
  http://{serverRoot}/push/{apiVersion}/clientCapabilities/{address}

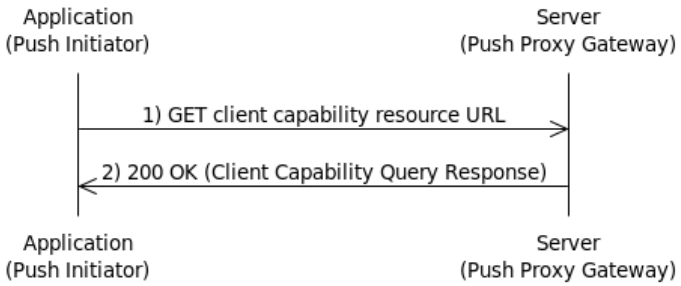

Figure 8: Query client capabilities by Push Initiator

1. The Push Initiator queries the capabilities of the client identified by an *address*, using GET.
2. The PPG responds with the capabilities of the client.

6. Detailed specification of the resources

6.1 Resource: Push messages

The resource used is:
http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}

This resource represents a Push message which can be created, replaced, and cancelled.

6.1.1 Request URI variables

The following request URI variables are common for all HTTP commands for this resource:

Name	Description
serverRoot	server base url: hostname+port+base path. Example: ppg.example.com/ExampleAPI
apiVersion	version of the Push Network API PI wants to use. The value of this variable is defined in section 5.1.
initiatorAddress	Unique string identifying the PI, as prior arranged between the PI and PPG via unspecified methods. MUST be URI-encoded if necessary per [RFC2396].
pushId	PI-assigned unique identifier (within the initiatorAddress scope) for the Push message. MUST be URI-encoded if necessary per [RFC2396].

Table 24 Push messages request URI variables

6.1.2 Response Codes and Error Handling

For HTTP response codes, see [REST_TS_Common].

For status codes, see [PAP].

For a mapping between both, see section 7.

6.1.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: PUT, DELETE' field in the response as per section 14.7 of [RFC 2616].

6.1.4 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: PUT, DELETE' field in the response as per section 14.7 of [RFC 2616].

6.1.5 PUT

This operation is used for Push message initiation and Push message replacement. If no resource exists at the URL against which the PUT method is invoked, a new Push message is initiated, addressed by that URL. If a resource already exists at the URL against which the PUT method is invoked, this Push message is replaced by the one passed in the entity body of the PUT request.

The HTTP return code reflects whether a new resource has been created (201 Created) or an existing resource has been replaced (200 OK).

Messages will remain on the server as described by [PPGService] unless cancelled by executing the DELETE command (see next section).

6.1.5.1 Example 1: Creation of a new Push message (Informative)

6.1.5.1.1 Request

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/xml"
Accept: application/xml
Content-Length: nnnn

--xj987hc
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<push-message xmlns="urn:oma:xml:rest:netapi:push:1"
  deliver-before-timestamp="2010-11-08T18:13:51.0Z"
  source-reference="source-reference1"
  ppg-notify-requested-to="http://pi1.example.com/Push/notify123"
  progress-notes-requested="true">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
  <address address-value="wappush=mary/type=user@ppg.example.com"/>
  <address address-value="wappush=alice/type=user@ppg.example.com"/>
  <quality-of-service priority="medium"/>
</push-message>

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/xml
Location: http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<push-response xmlns="urn:oma:xml:rest:netapi:push:1"
  sender-address="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
  sender-name="Push Gateway" reply-time="2010-11-08T18:13:51.0Z">
  <progress-note stage="stage1" note="note1" time="2010-11-08T18:13:51.0Z"/>
  <response-result code="1001" desc="The request has been accepted for processing"/>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</push-response>
```

6.1.5.2 Example 2: Replacement of an existing Push message (Informative)

6.1.5.2.1 Request

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
```

```
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/xml"
Accept: application/xml
Content-Length: nnnn

--xj987hc
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<push-message xmlns="urn:oma:xml:rest:netapi:push:1"
  replace-push-message="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
  replace-method="all"
  deliver-before-timestamp="2010-11-08T18:13:51.0Z"
  source-reference="source-reference1"
  ppg-notify-requested-to="http://pi1.example.com/Push/notify123"
  progress-notes-requested="true">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
  <address address-value="wappush=mary/type=user@ppg.example.com"/>
  <address address-value="wappush=alice/type=user@ppg.example.com"/>
  <quality-of-service priority="medium"/>
</push-message>

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

6.1.5.2.2 Response

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<push-response xmlns="urn:oma:xml:rest:netapi:push:1"
  sender-address="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
  sender-name="Push Gateway" reply-time="2010-11-08T18:13:51.0Z">
  <progress-note stage="stage1" note="note1" time="2010-11-08T18:13:51.0Z"/>
  <response-result code="1001" desc="The request has been accepted for processing"/>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</push-response>
```

6.1.5.3 Example 3: Syntax error (Informative)

The example passes the invalid value “some” in “replace-method”. Note that the XML instance below is not valid w.r.t. the Push Network API XML schema [SUP_NetAPI_Push]. A badmessage-response is generated as response.

6.1.5.3.1 Request

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/xml"
Accept: application/xml
Content-Length: nnnn
```

```
--xj987hc
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<push-message xmlns="urn:oma:xml:rest:netapi:push:1"
  replace-push-message="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
  replace-method="some"
  deliver-before-timestamp="2010-11-08T18:13:51.0Z"
  source-reference="source-reference1"
  ppg-notify-requested-to="http://pi1.example.com/Push/notify123"
  progress-notes-requested="true">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
  <address address-value="wappush=mary/type=user@ppg.example.com"/>
  <address address-value="wappush=alice/type=user@ppg.example.com"/>
  <quality-of-service priority="medium"/>
</push-message>

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

6.1.5.3.2 Response

```
HTTP/1.1 400 Bad Request
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<badmessage-response
    code="2000"
    desc="Syntax error: XML Syntax violated. Attribute (replace-method) with value (some) must have a value from the list (pending-only, all)"/>
```

6.1.6 DELETE

This operation is used for the cancellation of a whole Push message.

Note that an alternative way exists that allows partial cancellation, as described in section 6.3.

6.1.6.1 Example 1: Cancellation of a Push message (Informative)

6.1.6.1.1 Request

```
DELETE /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.1.6.1.2 Response

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:09 GMT
```

```
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cancel-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <cancel-result code="1000" desc="OK"/>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</cancel-response>
```

6.2 Resource: Push message delivery status

The resource used is:
http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/requests/{pushId}/status

This resource is for retrieving the status of a Push message.

6.2.1 Request URI variables

The following request URI variables are common for all HTTP commands for this resource:

Name	Description
serverRoot	server base url: hostname+port+base path. Example: ppg.example.com/ExampleAPI
apiVersion	version of the Push Network API PI wants to use. The value of this variable is defined in section 5.1.
initiatorAddress	Unique string identifying the PI, as prior arranged between the PI and PPG via unspecified methods. MUST be URI-encoded if necessary per [RFC2396].
pushId	PI-assigned unique identifier (within the initiatorAddress scope) for the Push message. MUST be URI-encoded if necessary per [RFC2396].

Table 25 Push message delivery status request URI variables

6.2.2 Response Codes and Error Handling

For HTTP response codes, see [REST_TS_Common].

For status codes, see [PAP].

For a mapping between both, see section 7.

6.2.3 GET

This operation is used for retrieving the status of a Push message.

Request URL parameters are:

Name	Type/value	Optional	Description
address	xsd:string [0..unbounded]	Yes	If given, this parameter defines the addresses for which the status is queried. If omitted, the query applies to all addresses of that particular Push message. MUST be URI-encoded per [RFC2396] if necessary .

Table 26 Push message status request URI variables

6.2.3.1 Example 1: Retrieval of the status of a Push message (Informative)

6.2.3.1.1 Request

```
GET /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<statusquery-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <statusquery-result event-time="2010-11-08T18:14:51.0Z" message-state="delivered" code="1000" desc="OK">
    <address address-value="wappush=bob/type=user@ppg.example.com"/>
    <quality-of-service priority="medium"/>
  </statusquery-result>
  <statusquery-result event-time="2010-11-08T18:14:51.0Z" message-state="pending" code="1001" desc="Accepted">
    <address address-value="wappush=mary/type=user@ppg.example.com"/>
    <quality-of-service priority="medium"/>
  </statusquery-result>
  <statusquery-result event-time="2010-11-08T18:14:51.0Z" message-state="rejected" code="2002" desc="Address Error">
    <address address-value="wappush=alice/type=user@ppg.example.com"/>
    <quality-of-service priority="medium"/>
  </statusquery-result>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</statusquery-response>
```

6.2.3.2 Example 2: Retrieval of the status of a Push message for a dedicated address (Informative)

6.2.3.2.1 Request

```
GET
/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status?address=wappush%3D12345%2Ftype%3Duser1%40ppg%2Eexa
mple%2Ecom HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.2.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<statusquery-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <statusquery-result event-time="2010-11-08T18:14:51.0Z" message-state="delivered" code="1000" desc="OK">
    <address address-value="wappush=bob/type=user@ppg.example.com"/>
    <quality-of-service priority="medium"/>
  </statusquery-result>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</statusquery-response>
```

6.2.3.3 Example 3: Request with invalid pushId (Informative)

6.2.3.3.1 Request

```
GET /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.2.3.3.2 Response

```
HTTP/1.1 404 Not Found
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<statusquery-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <statusquery-result code="2004" desc="Push ID Not Found" message-state="undeliverable"/>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</statusquery-response>
```

6.2.4 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.2.5 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.2.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.3 Resource: Partial Push message cancellation

The resource used is:

http://{serverRoot}/push/{apiVersion}/{initiatorAddress}/pushMessages/{pushId}/cancel

This resource is used for partially cancelling a Push message.

6.3.1 Request URI variables

The following request URI variables are common for all HTTP commands for this resource:

Name	Description
serverRoot	server base url: hostname+port+base path. Example: ppg.example.com/ExampleAPI
apiVersion	version of the Push Network API PI wants to use. The value of this variable is defined in section 5.1.
initiatorAddress	Unique string identifying the PI, as prior arranged between the PI and PPG via unspecified methods. MUST be URI-encoded per [RFC2396] if necessary.
pushId	PI-assigned unique identifier (within the initiatorAddress scope) for the Push message. MUST be URI-encoded per [RFC2396] if necessary.

Table 27 Partial Push message cancellation request URI variables

6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_TS_Common].

For status codes, see [PAP].

For a mapping between both, 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 [RFC 2616].

6.3.4 POST

This operation is used to partially cancel a Push message, i.e. to cancel it for some addresses. The list if the addresses to which the cancellation applies is specified in the entity body of the POST request.

6.3.4.1 Examples 1: Successful partial cancellation (Informative)

6.3.4.1.1 Request

```
POST /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/cancel HTTP/1.1
Host: ppg.example.com
Content-Type: application/xml
Accept: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cancel-message xmlns="urn:oma:xml:rest:netapi:push:1">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
</cancel-message>
```

6.3.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cancel-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <cancel-result code="1000" desc="OK">
    <address address-value="wappush=bob/type=user@ppg.example.com"/>
  </cancel-result>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</cancel-response>
```

6.3.4.2 Example 2: Unsuccessful partial cancellation (Informative)

6.3.4.2.1 Request

```
POST /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/cancel HTTP/1.1
Host: ppg.example.com
Content-Type: application/xml
Accept: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cancel-message xmlns="urn:oma:xml:rest:netapi:push:1">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
</cancel-message>
```

6.3.4.2.2 Response

```
HTTP/1.1 403 Forbidden
Date: Mon, 08 Nov 2010 18:14:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cancel-response xmlns="urn:oma:xml:rest:netapi:push:1">
  <cancel-result code="2008" desc="Cancellation not possible">
    <address address-value="wappush=bob/type=user@ppg.example.com"/>
  </cancel-result>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123</resourceURL>
</cancel-response>
```

Note: In this case, the HTTP error code can not be “404 Not Found”, as the underlying Push message resource exists, however the cancellation fails. Therefore, the error code chosen is “403 Forbidden”, which indicates according to [RFC2616] that “the server understood the request, but is refusing to fulfill it. […] If […] the server wishes to make public why the request has not been fulfilled, it SHOULD describe the reason for the refusal in the entity.”

6.3.5 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 [RFC 2616].

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 [RFC 2616].

6.4 Resource: Query client capabilities

The resource used is:
http://{serverRoot}/push/{apiVersion}/clientCapabilities/{address}?appId={appId}

This resource is for retrieving the capabilities of a Push client.

6.4.1 Request URI variables

The following request URI variables are common for all HTTP commands for this resource:

Name	Description
serverRoot	server base url: hostname+port+base path. Example: ppg.example.com/ExampleAPI
apiVersion	version of the Push Network API PI wants to use. The value of this variable is defined in section 5.1.
address	Client address whose capabilities are being queried. MUST be URI-encoded per [RFC2396] if necessary.
appId	This parameter contains the app-id attribute as defined by [PAP]. MUST be URI-encoded if necessary per [RFC2396].

Table 28Client Capability Query request URI variables

Note that the "query-id" attribute from [PAP] is not instantiated because it is only used for correlating request and response which is not needed in the Push Network API.

6.4.2 Response Codes and Error Handling

For HTTP response codes, see [REST_TS_Common].

For status codes, see [PAP].

For a mapping between both, see section 7.

6.4.3 GET

This operation is used the capabilities of a Push client.

6.4.3.1 Examples 1: Successful capability query (Informative)

6.4.3.1.1 Request

```
GET /ExampleAPI/push/v1/clientCapabilities/wappush%3Dbob%2Ftype%3Duser%40ppg.example.com?appId=urn%3Ax-wap-application%3Awml.ua HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.4.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: multipart/related; boundary=xj987hc; type="application/xml"
Content-Length: nnnn

--xj987hc
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<ccq-response xmlns="urn:oma:xml:rest:netapi:push:1" code="1000" desc="OK">
  <address address-value="wappush=bob/type=user@ppg.example.com"/>
   <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/clientCapabilities/wappush%3Dbob%2Ftype%3Duser%40ppg.example.com
?appId=urn%3Ax-wap-application%3Awml.ua
  </resourceURL>
</ccq-response>

--xj987hc
 Content-Type: application/xml

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#">
    <!--WAP Browser vendor site: Default description of WAP properties-->
    <rdf:Description ID="MyDeviceProfile">
        <prf:component>
            <rdf:Description ID="WAPProfile">
                <rdf:type resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#WapCharacteristics"/>
                <prf:WapVersion>2.0</prf:WapVersion>
                <prf:WmlDeckSize>1400</prf:WmlDeckSize>
                <prf:WapDeviceClass>A</prf:WapDeviceClass>
                <prf:WmlVersion>
                    <rdf:Bag>
                        <rdf:li>2.0</rdf:li>
                    </rdf:Bag>
                </prf:WmlVersion>
            </rdf:Description>
        </prf:component>
        <prf:component>
            <rdf:Description ID=":PushProfile">
                <rdf:type resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#PushCharacteristics"/>
                <prf:Push-Accept>
                    <rdf:Bag>
                        <rdf:li>text/vnd.wap.si</rdf:li>
                        <rdf:li>application/vnd.wap.sic</rdf:li>
                    </rdf:Bag>
                </prf:Push-Accept>
                <prf:Push-Accept-Language>
                    <rdf:Bag>
                        <rdf:li>en</rdf:li>
                    </rdf:Bag>
                </prf:Push-Accept-Language>
            </rdf:Description>
        </prf:component>
    </rdf:Description>
</rdf:RDF>
```

```
--xj987hc--
```

6.4.3.2 Example 2: Request with invalid client address (Informative)

6.4.3.2.1 Request

```
GET /ExampleAPI/push/v1/clientCapabilities/foobar?appId=urn%3Ax-wap-application%3Awml.ua HTTP/1.1
Host: ppg.example.com
Accept: application/xml
```

6.4.3.2.2 Response

```
HTTP/1.1 404 Not Found
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0"?>
<ccq-response xmlns="urn:oma:xml:rest:netapi:push:1"
  code="2003" desc="Address not found">  <address address-value="foobar"/>
  <resourceURL>http://ppg.example.com/ExampleAPI/push/v1/clientCapabilities/foobar?appId=urn%3Ax-wap-application%3Awml.ua
  </resourceURL>
</ccq-response>
```

6.4.4 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.4.5 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.4.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 14.7 of [RFC 2616].

6.5 Resource: PI notification about Push message delivery status

This resource is a client provided callback URL for notification about Push message delivery. The URL is provided by the PI to the PPG when a Push message is created or replaced. The Push Network API does not make any assumption about the structure of this URL.

6.5.1 Request URI variables

Client-provided if any.

6.5.2 Response Codes and Error Handling

For HTTP response codes, see [REST_TS_Common].

For status codes, see [PAP].

For a mapping between both, see section 7.

6.5.3 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 [RFC 2616].

6.5.4 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 [RFC 2616].

6.5.5 POST

This operation is used by the PPG to send a notification about the outcome of a submitted message for a specific recipient after the final result is known, in case the PI has requested to receive such notifications when creating or replacing a Push message.

The client MUST respond with a resultnotification-response instance in the entity body of the HTTP response.

6.5.5.1 Example: PI notification about the outcome of a Push message (Informative)

6.5.5.1.1 Request

```
POST /Push/notify123 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn
Host: pi1.example.com

<?xml version="1.0"?>
<resultnotification-message xmlns="urn:oma:xml:rest:netapi:push:1"
   sender-address="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
   sender-name="Push Gateway"
   received-time="2010-11-08T18:13:51.0Z" event-time="2010-11-08T18:14:12.0Z"
   message-state="delivered"
   code="1000" desc="OK">
  <address address-value="wappush=12345/type=user1@ppg.example.com"/>
  <link href="http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123" rel="push-message"/>
  <quality-of-service priority="medium"/>
</resultnotification-message>
```

6.5.5.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 08 Nov 2010 18:14:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<resultnotification-response xmlns="urn:oma:xml:rest:netapi:push:1"
  code="1000" desc="OK"/>
```

6.5.6 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 [RFC 2616].

7. PAP status code mapping to HTTP status codes

As noted in [PAP], "When using HTTP as a tunnel for PAP, the HTTP response codes are used only for HTTP layer conditions. All codes in PAP are conveyed through XML documents. When a PAP message has been accepted by the PPG or Push Initiator, the HTTP response code 202 is returned, even if the PAP message doesn't parse or is not well formed. Information on these failure conditions is returned in the response contained in the XML document."

In contrast, being a PAP binding to HTTP in the RESTful architectural style, the Push Network API relies upon HTTP status codes to express resource request status. The following table is adapted from [PAP], and extended with the applicable HTTP status code for the PAP response code. The HTTP status code mapping is intended to be aligned with the definitions in [RFC2616].

The table below lists the currently defined status codes and their meanings, and the HTTP status code to be provided.

			push-response	**cancel-response**	**statusquery-response**	**ccq-response**	**badmessage-response**
PAP Status Code	**Description**	**Interpretation**					
1000	OK	The request succeeded.		200	200	200	
1001	Accepted for Processing	The request has been accepted for processing.	201 (new) 200 (replaced)				
2000	Bad Request	Not understood due to malformed syntax.	400	400	400	400	400
2001	Forbidden	The request was refused.	403	403	403	403	
2002	Address Error	The client specified was not recognised.	400	400	400	400	
2003	Address Not Found	The address specified was not found.		404	404		
2004	Push ID Not Found	The Push ID specified was not found.	404	404	404		
2005	Capabilities Mismatch	The capabilities assumed by the PI were not acceptable for the client specified.	403				
2006	Required Capabilities Not Supported	The input is in a form not supported by the client.	403				
2007	Duplicate Push ID	The Push ID supplied is not unique within the PPG.	403 (occurs if replace not supported by PPG)				
2008	Cancellation not possible	The Push ID specified was found, but cancellation is not possible	403	403			
3000	Internal Server Error	Server could not fulfil request due to internal error.	500	500	500	500	
3001	Not Implemented	Server does not support the requested		500	500	500	

			push-response	cancel-response	statusquery-response	ccq-response	badmessage-response
PAP Status Code	**Description**	**Interpretation**					
		operation.					
3002	Version not Supported	The server refuses to support the protocol version indicated.					500
3003	Not Possible	Action not possible because message is no longer available.		410	410		
3004	Capability Matching not Supported	The PPG does not support client capability information provided in a push message.	500				
3005	Multiple Addresses Not Supported	The PPG does not support an operation that specified multiple recipients.	500	500	500		
3006	Transformation Failure	The PPG was unable to perform a transformation on the message.	500		500		
3007	Specified Delivery Method Not Possible	The PPG could not perform the confirmed or unconfirmed delivery specified.	500		500		
3008	Capabilities Not Available	Client capabilities for the specified client are not available.				404	
3009	Required Network Not Available	The network requested is not available.	403		403		
3010	Required Bearer Not Available	The bearer requested is not available.	403		403		
3011	Replacement Not Supported	The PPG does not support the replace operation	500				
3012	One-shot Not Supported	The PPG or the bearer does not support one-shot delivery.	500				
4000	Service Failure	The service failed. The client may re-attempt the operation.			500		
4001	Service Unavailable	The server is busy.			503		
5xxx	Mobile Client Aborted	The mobile client aborted the operation.			500		

Table 29 PAP Status Codes Mapped to HTTP Status Codes

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-REST_NetAPI_Push-V1_0-20131029-A	29 Oct 2013	Status changed to Approved by TP TP Ref # OMA-TP-2013-0335-INP_PushREST_V1_0_for_final_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for REST.Push Server

Item	Function	Reference	Requirement
REST-PUSH-SUPPORT-S-001-M	Support for PushREST API	5	
REST-PUSH-SUPPORT-S-002-M	Support for the XML request & response format	5	
REST-PUSH-SUPPORT-S-003-O	Support for the JSON request & response format	5, Appendix C	

B.1.1 SCR for REST.Push.PushMessage Server

Item	Function	Reference	Requirement
REST-PUSH-MSG-S-001-M	Support for Push messages	6.1	
REST-PUSH-MSG-S-002-M	Submit a new Push message - PUT	6.1.5	
REST-PUSH-MSG-S-003-M	Replace an existing Push message –PUT	6.1.5	
REST-PUSH-MSG-S-004-O	Cancel a Push message - DELETE	6.1.6	

B.1.2 SCR for REST.Push.StatusQuery Server

Item	Function	Reference	Requirement
REST-PUSH-SQ-S-001-O	Support for retrieval of a Push message delivery status	6.2	REST-PUSH-SQ-S-002-O
REST-PUSH-SQ-S-002-O	Query a Push message delivery status – GET	6.2.3	

B.1.3 SCR for REST.Push.PartialCancellation Server

Item	Function	Reference	Requirement
REST-PUSH-PC-S-001-O	Support for partially cancellation of a Push message	6.3	REST-PUSH-PC-S-002-O
REST-PUSH-PC-S-002-O	Partially cancel a Push message -POST	6.3.4	

B.1.4 SCR for REST.Push.CapQuery Server

Item	Function	Reference	Requirement
REST-PUSH-CQ-S-001-O	Support for retrieving the capabilities of a Push client	6.4	REST-PUSH-CQ-S-002-O
REST-PUSH-CQ-S-002-O	Query the capabilities of a Push client - GET	6.4.3	

B.1.5 SCR for REST.Push.ResultNotification Server

Item	Function	Reference	Requirement
REST-PUSH-RN-S-001-M	Support for notification about the outcome of a Push message	6.5	
REST-PUSH-RN-S-002-M	Send notification about the outcome of a Push message -POST	6.5.5	

Appendix C. 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 or response for various operations using a JSON binding. The examples follow the XML to JSON serialization rules in [REST_TS_Common]. A JSON response can be obtained by following the content negotiation guidelines section of [REST_TS_Common].

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

C.1 Creation of a new Push message (section 6.1.5.1)

Request:

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/json"
Accept: application/json
Content-Length: nnnn

--xj987hc
Content-Type: application/json

{"push-message": {
   "address": [
      {"address-value": "wappush=bob/type=user@ppg.example.com"},
      {"address-value": "wappush=mary/type=user@ppg.example.com"},
      {"address-value": "wappush=alice/type=user@ppg.example.com"}
   ],
   "deliver-before-timestamp": "2010-11-08T18:13:51.0Z",
   "ppg-notify-requested-to": "http://pi1.example.com/Push/notify123",
   "progress-notes-requested": "true",
   "quality-of-service": {"priority": "medium"},
   "source-reference": "source-reference1"
}}

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

Response:

```
HTTP/1.1 201 Created
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/json
Location: http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123
Content-Length: nnnn

{"push-response": {
   "progress-note": {
```

```
      "note": "note1",
      "stage": "stage1",
      "time": "2010-11-08T18:13:51.0Z"
   },
   "reply-time": "2010-11-08T18:13:51.0Z",
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "response-result": {
      "code": "1001",
      "desc": "The request has been accepted for processing"
   },
   "sender-address": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "sender-name": "Push Gateway"
}}
```

C.2 Replacement of an existing Push message (section 6.1.5.2)

Request:

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/json"
Accept: application/json
Content-Length: nnnn

--xj987hc
Content-Type: application/json

{"push-message": {
   "address": [
      {"address-value": "wappush=bob/type=user@ppg.example.com"},
      {"address-value": "wappush=mary/type=user@ppg.example.com"},
      {"address-value": "wappush=alice/type=user@ppg.example.com"}
   ],
   "deliver-before-timestamp": "2010-11-08T18:13:51.0Z",
   "ppg-notify-requested-to": "http://pi1.example.com/Push/notify123",
   "progress-notes-requested": "true",
   "quality-of-service": {"priority": "medium"},
   "replace-method": "all",
   "replace-push-message": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "source-reference": "source-reference1"
}}

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/json
Content-Length: nnnn
```

```
{"push-response": {
   "progress-note": {
      "note": "note1",
      "stage": "stage1",
      "time": "2010-11-08T18:13:51.0Z"
   },
   "reply-time": "2010-11-08T18:13:51.0Z",
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "response-result": {
      "code": "1001",
      "desc": "The request has been accepted for processing"
   },
   "sender-address": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "sender-name": "Push Gateway"
}}
```


C.3 Syntax error (section 6.1.5.3)

Request:

```
PUT /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Content-Type: multipart/related; boundary=xj987hc; type="application/json"
Accept: application/json
Content-Length: nnnn

--xj987hc
Content-Type: application/json

{"push-message": {
   "address": [
      {"address-value": "wappush=bob/type=user@ppg.example.com"},
      {"address-value": "wappush=mary/type=user@ppg.example.com"},
      {"address-value": "wappush=alice/type=user@ppg.example.com"}
   ],
   "deliver-before-timestamp": "2010-11-08T18:13:51.0Z",
   "ppg-notify-requested-to": "http://pi1.example.com/Push/notify123",
   "progress-notes-requested": "true",
   "quality-of-service": {"priority": "medium"},
   "replace-method": "some",
   "replace-push-message": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "source-reference": "source-reference1"
}}

--xj987hc
Content-Type: text/plain
Text Message Goes Here.
--xj987hc--
```

Response:

```
HTTP/1.1 400 Bad Request
Date: Mon, 08 Nov 2010 18:14:03 GMT
Content-Type: application/json
```

```
Content-Length: nnnn

{"badmessage-response": {
    "code": "2000",
    "desc": "Syntax error: XML Syntax violated. Attribute (replace-method) with value (some) must have a value from the list (pending-only, all)"
}}
```

C.4 Cancellation of a Push message (section 6.1.6.1)

Request:

```
DELETE /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123 HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:09 GMT
Content-Type: application/json
Content-Length: nnnn

{"cancel-response": {
   "cancel-result": {
      "code": "1000",
      "desc": "OK"
   },
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
}}
```

C.5 Retrieval of the status of a Push message (section 6.2.3.1)

Request:

```
GET /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"statusquery-response": {
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "statusquery-result": [
      {
```

```
        "address": {"address-value": "wappush=bob/type=user@ppg.example.com"},
        "code": "1000",
        "desc": "OK",
        "event-time": "2010-11-08T18:14:51.0Z",
        "message-state": "delivered",
        "quality-of-service": {"priority": "medium"}
      },
      {
        "address": {"address-value": "wappush=mary/type=user@ppg.example.com"},
        "code": "1001",
        "desc": "Accepted",
        "event-time": "2010-11-08T18:14:51.0Z",
        "message-state": "pending",
        "quality-of-service": {"priority": "medium"}
      },
      {
        "address": {"address-value": "wappush=alice/type=user@ppg.example.com"},
        "code": "2002",
        "desc": "Address Error",
        "event-time": "2010-11-08T18:14:51.0Z",
        "message-state": "rejected",
        "quality-of-service": {"priority": "medium"}
      }
    ]
}}
```


C.6 Retrieval of the status of a Push message for a dedicated address (section 6.2.3.2)

Request:

```
GET /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status?address=
wappush%3D12345%2Ftype%3Duser1%40ppg%2Eexample%2Ecom HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"statusquery-response": {
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "statusquery-result": {
      "address": {"address-value": "wappush=bob/type=user@ppg.example.com"},
      "code": "1000",
      "desc": "OK",
      "event-time": "2010-11-08T18:14:51.0Z",
      "message-state": "delivered",
      "quality-of-service": {"priority": "medium"}
   }
}}
```

C.7 Request with invalid pushId (section 6.2.3.3)

Request:

```
GET /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/status HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 404 Not Found
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"statusquery-response": {
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "statusquery-result": {
      "code": "2004",
      "desc": "Push ID Not Found",
      "message-state": "undeliverable"
   }
}}
```

C.8 Successful partial cancellation (section 6.3.4.1)

Request:

```
POST /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/cancel HTTP/1.1
Host: ppg.example.com
Content-Type: application/json
Accept: application/json
Content-Length: nnnn
```

```
{"cancel-message": {"address": {"address-value": "wappush=bob/type=user@ppg.example.com"}}}
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 08 Nov 2010 18:14:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"cancel-response": {
   "cancel-result": {
      "address": {"address-value": "wappush=bob/type=user@ppg.example.com"},
      "code": "1000",
      "desc": "OK"
   },
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
}}
```

C.9 Unsuccessful partial cancellation (section 6.3.4.2)

Request:

```
POST /ExampleAPI/push/v1/pi1.example.com/pushMessages/id123/cancel HTTP/1.1
Host: ppg.example.com
Content-Type: application/json
Accept: application/json
Content-Length: nnnn

{"cancel-message": {"address": {"address-value": "wappush=bob/type=user@ppg.example.com"}}}
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Mon, 08 Nov 2010 18:14:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"cancel-response": {
   "cancel-result": {
      "address": {"address-value": "wappush=bob/type=user@ppg.example.com"},
      "code": "2008",
      "desc": "Cancellation not possible"
   },
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"
}}
```

C.10 Successful capability query (section 6.4.3.1)

Request:

```
GET /ExampleAPI/push/v1/clientCapabilities/wappush%3Dbob%2Ftype%3Duser%40ppg.example.com?appId=urn%3Ax-wap-application%3Awml.ua HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: multipart/related; boundary=xj987hc; type="application/json"
Content-Length: nnnn

--xj987hc
Content-Type: application/json

{"ccq-response": {
    "address": {"address-value": "wappush=bob/type=user@ppg.example.com"},
    "code": "1000",
   "desc": "OK"
   "resourceURL":
"http://ppg.example.com/ExampleAPI/push/v1/clientCapabilities/wappush%3Dbob%2Ftype%3Duser%40ppg.example.com?appId=urn%3Ax-wap-application%3Awml.ua"
}}

--xj987hc
 Content-Type: application/xml

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#">
    <!--WAP Browser vendor site: Default description of WAP properties-->
    <rdf:Description ID="MyDeviceProfile">
        <prf:component>
            <rdf:Description ID="WAPProfile">
                <rdf:type resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#WapCharacteristics"/>
                <prf:WapVersion>2.0</prf:WapVersion>
                <prf:WmlDeckSize>1400</prf:WmlDeckSize>
                <prf:WapDeviceClass>A</prf:WapDeviceClass>
                <prf:WmlVersion>
                    <rdf:Bag>
                        <rdf:li>2.0</rdf:li>
                    </rdf:Bag>
                </prf:WmlVersion>
            </rdf:Description>
        </prf:component>
        <prf:component>
            <rdf:Description ID=":PushProfile">
                <rdf:type resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#PushCharacteristics"/>
                <prf:Push-Accept>
                    <rdf:Bag>
                        <rdf:li>text/vnd.wap.si</rdf:li>
                        <rdf:li>application/vnd.wap.sic</rdf:li>
                    </rdf:Bag>
                </prf:Push-Accept>
```

```
            <prf:Push-Accept-Language>
                <rdf:Bag>
                    <rdf:li>en</rdf:li>
                </rdf:Bag>
            </prf:Push-Accept-Language>
        </rdf:Description>
    </prf:component>
  </rdf:Description>
</rdf:RDF>
--xj987hc--
```

C.11 Request with invalid client address (section 6.4.3.2)

Request:

```
GET /ExampleAPI/push/v1/clientCapabilities/foobar?appId=urn%3Ax-wap-application%3Awml.ua HTTP/1.1
Host: ppg.example.com
Accept: application/json
```

Response:

```
HTTP/1.1 404 Not Found
Date: Mon, 08 Nov 2010 18:17:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"ccq-response": {
   "address": {"address-value": "foobar"},
   "code": "2003",
   "desc": "Address not found"
   "resourceURL": "http://ppg.example.com/ExampleAPI/push/v1/clientCapabilities/foobar?appId=urn%3Ax-wap-application%3Awml.ua"
}}
```

C.12 PI notification about the outcome of a Push message (section 6.5.5.1)

Request:

```
POST /Push/notify123 HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: nnnn
Host: pi1.example.com

{"resultnotification-message": {
   "address": {"address-value": "wappush=12345/type=user1@ppg.example.com"},
   "code": "1000",
   "desc": "OK",
   "event-time": "2010-11-08T18:14:12.0Z",
   "link": {"href": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123"}, "rel": "push-message"},
   "message-state": "delivered",
```

```
   "quality-of-service": {"priority": "medium"},
   "received-time": "2010-11-08T18:13:51.0Z",
   "sender-address": "http://ppg.example.com/ExampleAPI/push/v1/pi1.example.com/pushMessages/id123",
   "sender-name": "Push Gateway"
}}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Mon, 08 Nov 2010 18:14:59 GMT

{"resultnotification-response": {
   "code": "1000",
   "desc": "OK"
}}
```