WAP Push Access Protocol

Proposed Version 16-August-1999

Wireless Application Protocol
Push Access Protocol Specification

Contents

1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and providing new services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation, and fast/flexible service creation, WAP defines a set of protocols in transport, session and application layers. For additional information on the WAP architecture, refer to "*Wireless Application Protocol Architecture Specification*" [WAP].

This specification defines the Push Access Protocol (PAP). The PAP is intended for use in delivering content from Push Initiators to push proxy gateways for subsequent delivery to narrow band devices, including cellular phones and pagers. Example messages include news, stock quotes, weather, traffic reports, and notification of events such as email arrival. With push functionality, users are able to get information without having to request that information. In many cases it is important for the user to get the information as soon as it is available.

The push access protocol is not intended for use over the air.

2. Document Status

This document is available online in the following formats:

PDF format at http://www.wapforum.org/.

2.1 Copyright Notice

2.2 Errata

Known problems associated with this document are published at http://www.wapforum.org/.

2.3 Comments

Comments regarding this document can be submitted to the WAP Forum in the manner published at http://www.wapforum.org/.

3. References

3.1 Normative References

[ISO8601]	"Data elements and interchange formats - Information interchange - Representation of dates and times", International Organization For Standardization (ISO), 15-June-1988 "Data elements and interchange formats - Information interchange - Representation of dates and times, Technical Corrigendum 1", International Organization For Standardization (ISO) - Technical Committee ISO/TC 154, 01-May-1991
[RFC1738]	"Uniform Resource Locators (URL)", T. Berners-Lee, et al., December 1994. URL: http://www.ietf.org/rfc/rfc1738.txt
[RFC2046]	"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed, et al. November 1996, URL: http://www.ietf.org/rfc/rfc2046.txt
[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
[RFC2616]	"Hypertext Transfer Protocol – HTTP/1.1", R. Fielding, et al. June 1999, URL:http://www.ietf.org/rfc/rfc2616.txt
[UAPROF]	"Wireless Application Group User Agent Profiles Specification", WAP Forum, 22-June-1999 URL: http://www.wapforum.org/
[WINA]	"WAP Interim Naming Authority", WAP Forum, URL:http://www.wapforum.org/wina/
[XML]	"Extensible Markup Language (XML)", W3C Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998. URL: http://www.w3.org/TR/REC-xml

3.2 Informative References

[PushArch]	"WAP Push Architectural Overview", WAP Forum, 16-August-1999 URL: http://www.wapforum.org/
[PushMsg]	"WAP Push Message", WAP Forum, 16-August-1999 URL: http://www.wapforum.org/
[RDF]	"Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation, 22-February-1999. URL: http://www.w3.org/TR/REC-rdf-syntax
[RFC822]	"Standard for the Format of ARPA Internet Text Messages", D. Crocker, August 1982. URL: http://www.ietf.org/rfc/rfc0822.txt
[RFC2246]	"The TLS Protocol Version 1.0", T. Dierks, C. Allen. January 1999, URL:http://www.ietf.org/rfc/rfc2246.txt
[RFC2396]	"Uniform Resource identifiers (URI)", T. Berners-Lee, et al., August 1998. URL: http://www.ietf.org/rfc/rfc2396.tx
[WAP]	"Wireless Application Protocol Architecture Specification", WAP Forum, 30-April-1998 URL: http://www.wapforum.org/

4. Definitions and Abbreviations

4.1 Definitions

The following are terms and conventions used throughout this specification.

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 by [RFC2119].

Application - A value-added data service provided to a WAP Client. The application may utilise both push and pull data transfer to deliver content

Application-Level Addressing - the ability to address push content between a particular user agent on a WAP client and push initiator on a server.

Bearer Network - a network used to carry the messages of a transport-layer protocol between physical devices. Multiple bearer networks may be used over the life of a single push session.

Client – in the context of push, a client is a device (or service) that expects to receive push content from a server. In the context of pull a client, it is a device initiates a request to a server for content or data. See also “device”.

Contact Point – address information that describes how to reach a push proxy gateway, including transport protocol addres and port of the push proxy gateway.

Content - subject matter (data) stored or generated at an origin server. Content is typically displayed or interpreted by a user agent on a client. Content can both be returned in response to a user request, or being pushed directly to a client.

Content Encoding - when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.

Content Format – actual representation of content.

Context – an execution space where variables, state and content are handled within a well-defined boundary.

Device – is a network entity that is capable of sending and/or receiving packets of information and has a unique device address. A device can act as either a client or a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.

End-user - see “user”

Extensible Markup Language - is a World Wide Web Consortium (W3C) recommended standard for Internet mark-up languages, of which WML is one such language. XML is a restricted subset of SGML.

Multicast Message - a push message containing a single OTA client address which implicitly specifies more than OTA client address.

Push Access Protocol - a protocol used for conveying content that should be pushed to a client, and push related control information, between a Push Initiator and a Push Proxy/Gateway.

Push Framework- - the entire WAP push system. The push framework encompasses the protocols, service interfaces, and software entities that provide the means to push data to user agents in the WAP client.

Push Initiator - the entity that originates push content and submits it to the push framework for delivery to a user agent on a client.

Push OTA Protocol - a protocol used for conveying content between a Push Proxy/Gateway and a certain user agent on a client.

Push Proxy Gateway - a proxy gateway that provides push proxy services.

Push Session - A WSP session that is capable of conducting push operations.

Server - a device (or service) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client. A server may initiate a connection to a client as part of a service (push).

User - a user is a person who interacts with a user agent to view, hear, or otherwise use a rendered content. Also referred to as end-user.

User agent - a user agent (or content interpreter) is any software or device that interprets resources. This may include textual browsers, voice browsers, search engines, etc.

XML – see *Extensible Markup Language*

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

CPI	Capability and Preference Information
DNS	Domain Name Server
DTD	Document Type Definition
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
OTA	Over The Air
PAP	Push Access Protocol
PI	Push Initiator
PPG	Push Proxy Gateway
QOS	Quality of Service
RDF	Resource Description Framework
RFC	Request For Comments
SGML	Standard Generalized Markup Language
SI	Service Indication
SIA	Session Initiation Application
SIR	Session Initiation Request
SL	Service Loading
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Co-ordinated
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WBXML	WAP Binary XML
WINA	WAP Interim Naming Authority
WTLS	Wireless Transport Layer Security
XML	Extensible Mark-up Language

5. Introduction

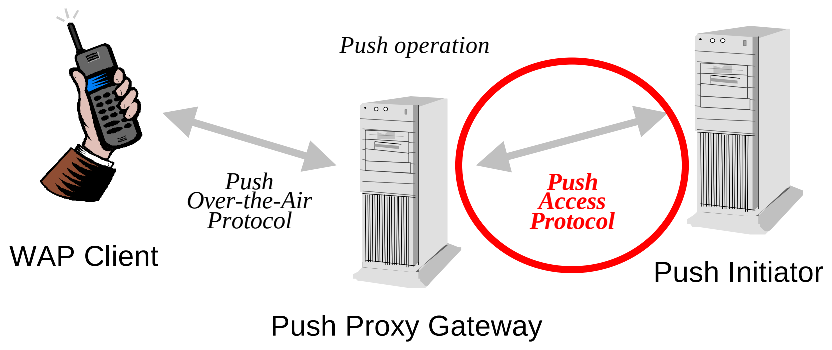

Figure 1. WAP Push Architecture

Figure 1 shows the WAP Push architecture. The *Push Access* protocol is used by a Push Initiator residing on an Internet server to access a push proxy gateway. This *access* protocol is the subject of this specification.

The Push Access Protocol is designed to be independent of the underlying transport protocol. This specification defines in section 14 how to implement PAP over HTTP [RFC2616]; other transports (e.g. SMTP) may be described in the future.

6. Operations

This section gives an overview of the operations defined for the push access protocol. Detailed field definitions can be found in section 9.

The Push Initiator is able to initiate the following operations to the Push Proxy Gateway:

a) Submit a Push (see section 6.1)

b) Cancel a Push (see section 6.3)

c) Query for status of a Push (see section 6.4)

d) Query for wireless device capabilities (see section 6.5)

The PPG is able to initiate the following message to the Push Initiator:

a) Result notification (see section 6.2)

All operations are request/response - for every initiated message, there is a response back to the initiator.

6.1 Push Submission

The purpose of the Push Submission is to deliver a push message from a Push Initiator to a PPG, which should then deliver the message to a user agent in a device on the wireless network. The Push message contains a control entity and a content entity, and MAY contain a capabilities entity. The control entity is an XML document that contains control information (`push-message,` section 9.2) for the PPG to use in processing the message for delivery. The content entity represents content to be sent to the wireless device. The capabilities entity contains client capabilities assumed by the Push Initiator and is in the RDF [RDF] format as defined in the User Agent Profile [UAPROF]. The PPG MAY use the capabilities information to validate that the message is appropriate for the client.

The response to the push request is an XML document (`push-response,` section 9.3) that indicates initial acceptance or failure. At minimum the PPG MUST validate against the DTD [XML] the control entity in the message and report the result in the response. The PPG MAY indicate, using `progress-note` (if requested by the Push initiator in the `progress-notes-requested` attribute), that other validations have been completed. The contents and number of `progress-note`s are implementation specific. A typical response message may contain progress notes for each stage of internal processing. The processing stages used are implementation specific.

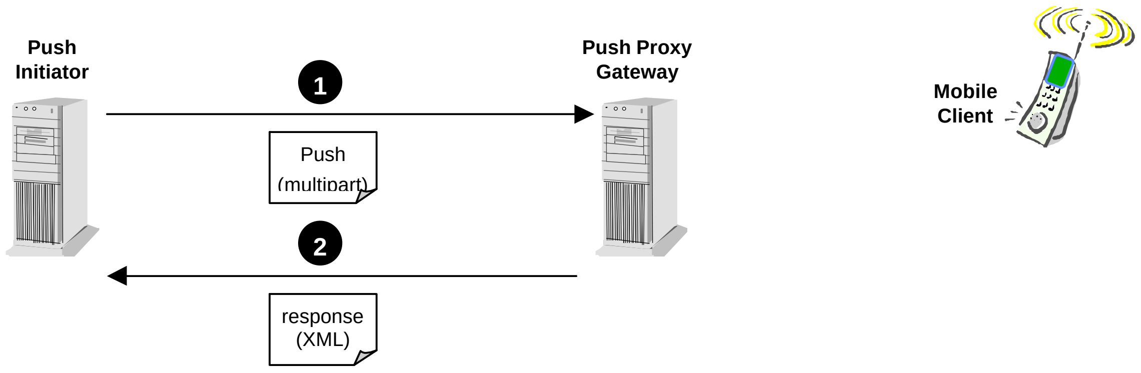

Figure 2. Push Submission

There are provisions in the Push message to specify multiple recipients. The response message corresponds to the submit message, so there is one response message for one push message, regardless of the number of addresses specified.

If the Push Initiator desires information related to the final outcome of the delivery, then it MUST request a result notification information in the push submission and provide a return address (e.g. URL).

Support for the push submission operation is REQUIRED in a PPG.

6.2 Result Notification

This operation is used by the PPG to inform the initiator of the final outcome of a push submission, if requested by the Push Initiator. This notification (arrow 5, below) tells the Push Initiator that the message was sent (transmitted, as in arrow 3), delivered (confirmation received from wireless device, as in arrow 4), it expired, was cancelled, or there was an error. If there was a processing error, the notification SHOULD be sent immediately upon detection of the error to the Push Initiator and the message should not be sent to the client. Otherwise, the notification MUST be sent after the message delivery process has been completed. The delivery process is considered completed when the message is no longer a candidate for delivery, e.g. the message has expired. If the push submission is indicated as rejected in step two in figure 3, then no result notification will be sent. The Push Initiator MUST have provided a return address (e.g. URL) during the push operation for this notification to be possible.

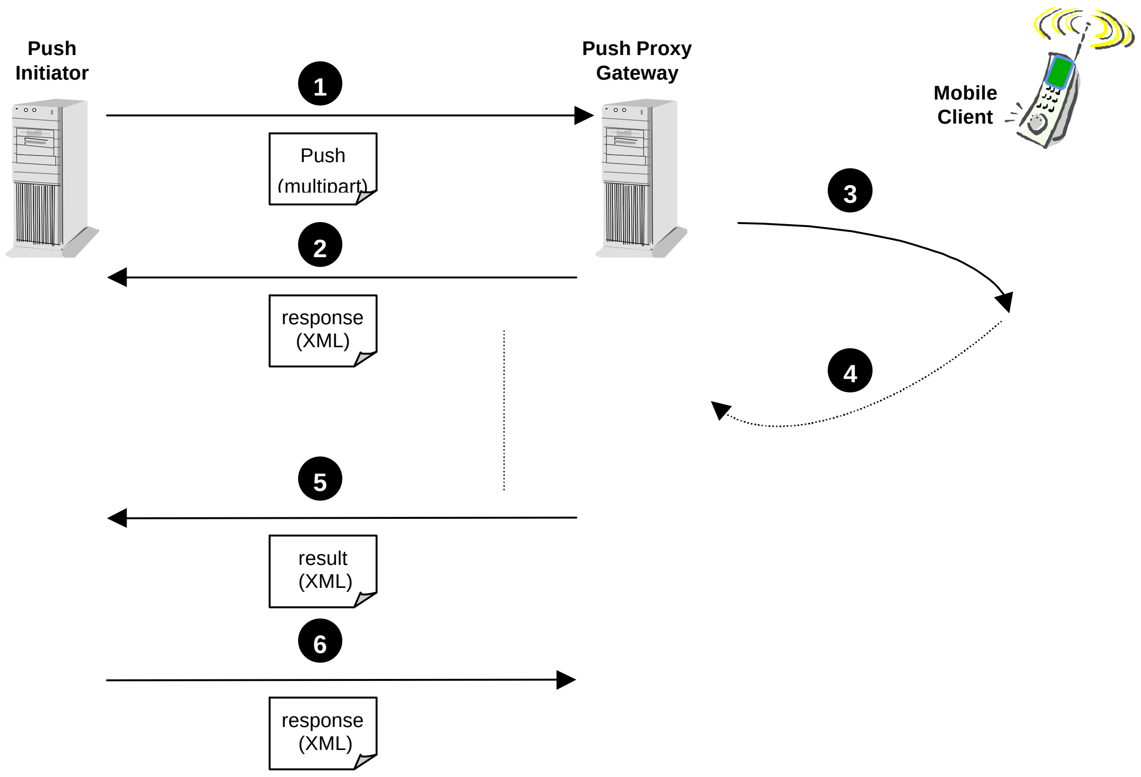

Figure 3. Push Submission with Confirmation

The `delivery-method` is reported as "`unconfirmed`" after unconfirmed delivery. The `delivery-method` is reported as "`confirmed`" after confirmed delivery.

The result notification message contains one entity - an XML document that contains the final message outcome (`resultnotification-message,` section 9.6). The response to the result notification message is also an XML document (`resultnotification-response,` section 9.7).

Support for this operation is REQUIRED in a PPG.

6.3 Push Cancellation

The purpose of the Push Cancellation is to allow the Push Initiator to attempt to cancel a previously submitted push message. The Push Initiator initiates this operation. The PPG responds with an indication of whether the request was successful or not.

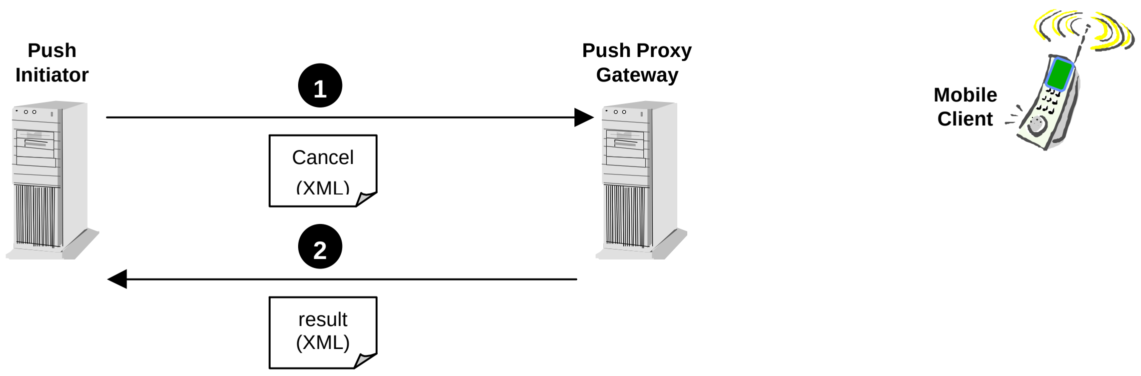

Figure 4. Push Cancellation

The cancel message (`cancel-message`, section 9.4) and the result message (`cancel-response`, section 9.5) are XML documents. If a message for which a result notification was requested is cancelled, the result notification MUST be sent and MUST report a `message-state` of "cancelled".

Support for this operation is OPTIONAL in a PPG. If this capability is not supported in a PPG, the status "Not Implemented" MUST be returned in the response.

6.4 Status Query

The status query operation allows the Push Initiator to request the current status of a message that has been previously submitted. If status is requested for a message which is addressed to multiple recipients, the PPG MUST send back a single response containing status query results for each of the recipients.

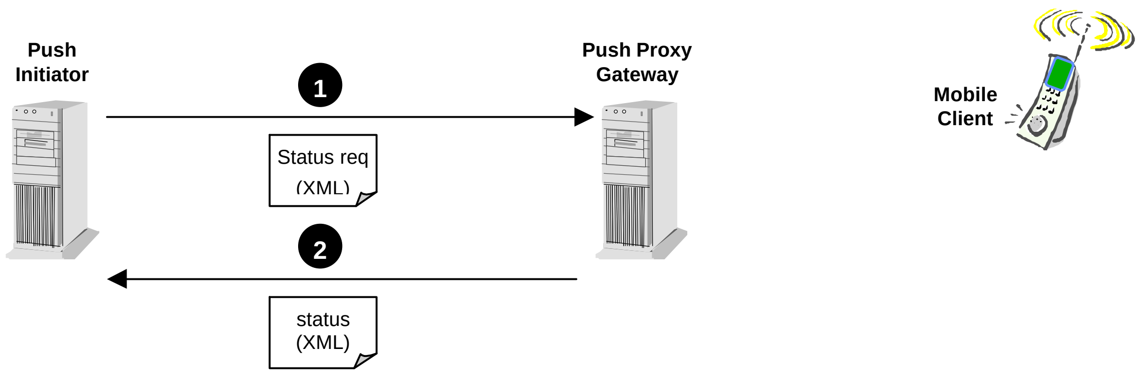

Figure 5. Status Query Operation

The status request (`statusquery-message`, section 9.8) and response (`statusquery-response`, section 9.9) messages are XML documents.

Support for this operation is OPTIONAL in a PPG. If this capability is not supported in a PPG, the status "Not Implemented" MUST be returned in the response.

6.5 Client Capabilities Query

This operation allows the Push Initiator to query the PPG for the capabilities of a specific device. The response is a multipart/related document containing the `ccq-response` (section 9.11) element in an XML document and, in the second entity, the actual client capabilities information in RDF [RDF] as defined in the User Agent Profile [UAPROF]. The PPG MAY add to the capabilities reported if the PPG is willing to perform transformations to the formats supported by the client. For example, if a client has JPG support but not GIF and a PPG is willing to convert GIF files to JPG, then the PPG may report that the client can support JPG and GIF files. The capabilities reported may be the combined PPG and client capabilities and they may have been derived from session capabilities or retrieved from a CC/PP server. Capabilities may also be derived using implementation dependent means.

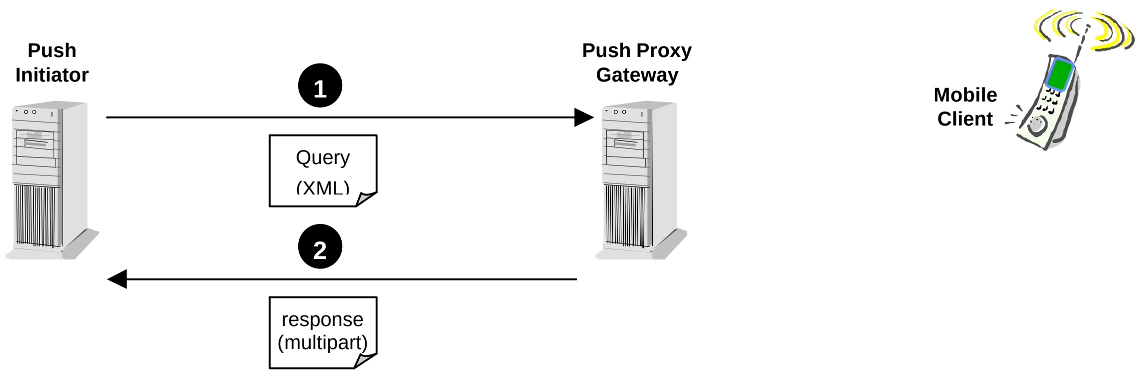

Figure 6. Client Capabilities Query Operation

The query message (`ccq-message`, section 9.10) is an XML document that specifies the client for which the capabilities are desired.

Support for this operation is OPTIONAL in a PPG.

7. Addressing

There are three addresses to be considered by the Push Initiator: the push proxy gateway address, the wireless device address, and the result notification address. The push proxy gateway address must be known by the Push Initiator. This address is needed at the layer below the push access protocol. The push proxy gateway is addressed using a unique address that depends on the underlying protocol. For example, when the underlying protocol is HTTP, a URL [RFC1738] is used.

The device addressing information is included as part of the message content (XML tagged content). Any character allowed in an RFC822 address may appear in the device address field.

In addition, a "`notify-requested-to`" address may be provided by the Push Initiator when required so that the push proxy gateway can later respond to the Push Initiator with result notification.

7.1 Multiple Recipient Addressing

There are scenarios in which a Push Initiator may want to send identical messages to multiple recipients. Rather than submitting multiple identical push messages, one to each recipient, the Push Initiator may submit a single push message addressed to multiple recipients. This section is intended to clarify behaviour related to operations on multiple recipients.

When the PPG returns the `push-response` message, after a push submission to multiple recipients, the response corresponds to the message, regardless of the number of recipients specified in the push submission (there is one response for each push submission).

When a Push Initiator requests status (section 9.8) with multiple addresses specified, the PPG MUST reply with a single statusquery-response (section 9.9) containing the individual statuses. The same is true when only a `push-id` is specified (no address specified) in the query for status of a multiple recipient message.

Result notifications (section 9.6) MUST be sent by the PPG for each individual recipient, if result notification is requested by the Push Initiator during the submission of a message to multiple recipients.

In cases where a message is sent to multiple recipients and later a cancel is requested by the initiator, the PPG MAY send back individual responses related to each of the multiple recipients or it MAY send responses related to many or all of the recipients.

Support of multiple addresses is OPTIONAL in a PPG.

7.2 Multicast/Broadcast Addresses

There are scenarios in which a single address submitted by a PI may be expanded by a PPG into multiple addresses for delivery. In addition, a single address transmitted on a wireless network may be received by multiple devices (e.g. broadcast). This type of service is expected for the distribution of information of interest to a broad population (e.g. news, weather, and traffic). This section is intended to clarify behaviour related to operations involving multicast and broadcast addresses.

Since the address expansion is done in the PPG or in the wireless network, the behaviour between the PI and the PPG is identical to behaviour as if the address were not expanded. The response contains the individual address as submitted by the PI.

8. Message Format

The push access protocol is independent of the transport used.

PAP messages carry control information, and in the case of a push submission, also content and optionally client capabilities information. Control information includes command/response messages between the PPG and the Push Initiator, and parameters passed to the PPG for use in sending content to the wireless device. Examples of this type of information include the wireless device address, the delivery priority of the message, etc. This information is not normally delivered to the wireless device. Content is information that is intended for the wireless device. This information might be intelligible only to the wireless device (e.g. may be encrypted by the Push Initiator or may be application data for an application unknown to the PPG) or it may be recognisable by the PPG (e.g. HTML or WML). The PPG may be configured to perform some transformation on recognisable content (e.g. HTML to WML) for certain wireless devices. The other category of information is client capability information as specified in the User Agent Profile [UAPROF].

When more than control is carried in a message, the format of the message is a MIME multipart/related [RFC2387] compound object. When only control information (e.g. for message responses) is carried in a message, the format of the message is a simple application/xml entity.

All information is transported within a single message body. In the multipart messages, the first entity contains all push related control information in an XML document, the second entity contains the content for the wireless device, the third entity, if present, contains UAPROF client capabilities. The format of the content entity is specified in [PushMsg].

8.1 Control Entity Format

The control entity is a MIME body part which holds an XML document containing one `pap` element as defined in section 9.1. The control entity MUST be included in every PAP request and response. The control entity MUST be the first entity in the MIME multipart/related message.

8.2 Content Entity Format

The content entity is a MIME body part containing the content to be sent to the wireless device. The content type is not defined by the PAP, but can be any type as long as it is described by MIME. The content entity is included only in the push submission and is not included in any other operation request or response. The content entity MUST be the second entity in the MIME multipart/related message.

8.3 Capabilities Entity Format

The capabilities entity is a MIME body part containing the Push Initiator's assumed subset of the capabilities of the wireless device/user agent. The capabilities format is specified in the User Agent Profile [UAPROF]. The capabilities entity, if present, MUST be the third entity in the Push Submission MIME multipart/related message and MUST be the second entity in a Client Capabilities Query response.

8.4 Example

The following is an example of a push request multipart body containing an XML document (control entity), a content entity, and a capabilities entity.

```
Content-Type: multipart/related; boundary=asdlfkjiurwghasf;
      type="application/xml"

--asdlfkjiurwghasf
```

```
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>
  ..control for PPG..
</pap>

--asdlfkjiurwghasf
Content-Type: text/vnd.wap.wml

  ..message content..

--asdlfkjiurwghasf
Content-Type: application/xml

  ..assumed client capabilities..

--asdlfkjiurwghasf--
```

The XML document contains all control information needed by the PPG to deliver the message to the wireless client. The information intended for the wireless client is carried in the second entity as a push message [PushMsg].

9. Control Elements & Attributes

This section describes the elements found in the DTD. They are described in the order they appear in the DTD.

9.1 The pap Element

```
<!ELEMENT pap                    ( push-message
                                 | push-response
                                 | cancel-message
                                 | cancel-response
                                 | resultnotification-message
                                 | resultnotification-response
                                 | statusquery-message
                                 | statusquery-response
                                 | ccq-message
                                 | ccq-response
                                 | badmessage-response) >
<!ATTLIST pap
          product-name           CDATA            #IMPLIED
>
```

The `pap` element specifies a set of elements that describe messages. The elements are tagged with "message" or "response" suffix to make it clear when they are messages or responses to messages.

Attributes

`product-name=CDATA`

This attribute contains the name or identification of the `pap` handling application that generated the message. It may be useful in operational compatibility between various vendor implementations.

9.2 The push-message Element

```
<!ELEMENT push-message ( address+, quality-of-service? ) >
<!ATTLIST push-message
          push-id                    CDATA              #REQUIRED
          deliver-before-timestamp   %Datetime;         #IMPLIED
          deliver-after-timestamp    %Datetime;         #IMPLIED
          source-reference           CDATA              #IMPLIED
          ppg-notify-requested-to    CDATA              #IMPLIED
          progress-notes-requested   ( true | false )   "false"
>
```

The `push-message` element has two component elements that further describe push messages. The PPG MUST support one address and MAY support multiple addresses (`address+`).

Direction: (Push Initiator→PPG)

Attributes

`push-id=CDATA`

This attribute is assigned by the Push Initiator and serves as a message ID. It can be used to cancel or check status on this message. The Push Initiator is responsible for its global uniqueness.

In order avoid conflicts between `push-id`s, it is recommended that content developers use an address (e.g. URL) within their control combined with an identifier for the `push-message` as the value for `push-id` (for example: “`www.wapforum.org/123`” or “`123@wapforum.org`”).

`deliver-before-timestamp=%Datetime`

This attribute specifies the date and time by which the content must be delivered to the wireless device. Content that has aged beyond this date MUST not be transmitted. PPGs that do not support this function MUST reject the message. The time MUST be represented in Co-ordinated Universal Time (UTC), a 24-hour timekeeping system. The following representation based on [ISO8601] MUST be used:

`YYYY-MM-DDThh:mm:ssZ`

Where:	`YYYY`	= 4 digit year (“0000” ... “9999”)
	`MM`	= 2 digit month (“01”=January, “02”=February ... “12”=December)
	`DD`	= 2 digit day (“01”, “02” ... “31”)
	`hh`	= 2 digit hour, 24-hour timekeeping system (“00” ... “23”)
	`mm`	= 2 digit minute (“00” ... “59”)
	`ss`	= 2 digit second (“00” ... “59”)
	`Z`	= UTC

Example: “`1999-04-30T06:45:00Z`” means 6.45 in the morning on the 30th of April 1999 UTC.

`deliver-after-timestamp=%Datetime`

This attribute specifies the date and time after which the content should be delivered to the wireless device. Content MUST not be transmitted before this date. The time format is the same as that used in `deliver-before-timestamp.` PPGs that do not support this function MUST reject the message.

`source-reference=CDATA`

This attribute contains a textual name of the content provider. This is useful to a PPG operator in identifying the originator of the message.

`ppg-notify-requested-to=CDATA`

This attribute specifies the address (e.g. URL) that the PPG should use for notification of results, using `resultnotification-message`, related to this message. The presence of this attribute indicates that the notification is requested. If the Push Initiator does not want a notification, then the Push Initiator MUST NOT provide this attribute. If a notification is requested, the PPG MUST later indicate the message outcome to the Push Initiator at the address specified in this attribute.

`progress-notes-requested=( true | false )`

This attribute informs the PPG as to whether or not the PI wants to receive progress notes. A value of "`true`" means that notes are requested.

9.2.1 The address Element

```
<!ELEMENT address EMPTY >
<!ATTLIST address
          address-value          CDATA               #REQUIRED
>
```

The address element contains the target device address for use by the PPG.

Attributes

`address-value=CDATA`

This attribute must contain a text string that represents the client address. It may be a logical address.

9.2.2 The quality-of-service Element

```
<!ELEMENT quality-of-service EMPTY >
<!ATTLIST quality-of-service
          priority              ( high | medium | low )        "medium"
          delivery-method       ( confirmed | preferconfirmed
                                | unconfirmed | notspecified ) "notspecified"
          network               CDATA                          #IMPLIED
          network-required      ( true | false )               "false"
          bearer                CDATA                          #IMPLIED
          bearer-required       ( true | false )               "false"
>
```

The `quality-of-service` element conveys the delivery qualities desired by the Push Initiator. When the requested `quality-of-service` cannot be honoured by the PPG, the request MUST be rejected with an appropriate error code. It is also used to convey the delivery qualities that were used by the PPG during message delivery. When sent to the PI, it MUST reflect the delivery qualities that were used by the PPG.

Attributes

`priority=( high | medium | low )`

This attribute specifies the delivery priority of the message. Valid values are “`low`”, “`medium`”, and “`high`”. The value "`high`" indicates the fastest delivery is desired. The value "`low`" indicates the slowest delivery. The actual delivery latencies associated with these qualities are implementation specific. The methods used to reduce delivery latency are implementation dependent, but may include the use of a different bearer or the reordering of messages waiting transmission to send the higher priority messages first. PPG support of priority functionality is OPTIONAL.

`delivery-method=( confirmed | preferconfirmed | unconfirmed | notspecified )`

The `delivery-method` attribute is used to specify the over the air delivery desired by the Push Initiator. Valid values are “`confirmed`”, "`preferconfirmed`", "`unconfirmed`" and "`notspecified`". When `delivery-method` is "`confirmed`", the PPG MUST use confirmed delivery of the message to the client. Note that a Push Initiator may request client confirmation without requesting `ppg-confirm-request` - the result is that the message is confirmed over the air but the PPG does not inform the Push Initiator. The value "`preferconfirmed`" allows the Push Initiator to inform the PPG of preferences. The PPG SHOULD try to deliver the message as preferred, but may use another method if not able to use the preferred choice. The value "`unconfirmed`" means that the message MUST be delivered in an unconfirmed manner. The value "`notspecified`" indicates that the Push Initiator does not care whether the PPG uses confirmed delivery or unconfirmed delivery - the choice is up to the PPG.

`network=CDATA`

The network desired for use when delivering the message. Network types are registered with WINA [WINA]

`network-required=( true | false )`

If "true" the specified network must be used.

`bearer=CDATA`

The bearer desired for use when delivering the message. Bearer types are registered with WINA [WINA].

`bearer-required=( true | false )`

If "true" the specified bearer must be used

9.3 The push-response Element

```
<!ELEMENT push-response ( progress-note*, response-result ) >
<!ATTLIST push-response
          push-id               CDATA             #REQUIRED
          sender-address        CDATA             #IMPLIED
          sender-name           CDATA             #IMPLIED
          reply-time            %Datetime;        #IMPLIED
>
```

The `push-response` element contains information about the outcome of the push submission. The `push-id` can be used by the Push Initiator to associate the response with the related push-message.

Direction: (PPG→Push Initiator)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the corresponding `push-message`. It can be used to match the response to the message.

`sender-address=CDATA`

This attribute contains the address to which the message was originally posted (e.g. the PPG's URL in the case of a message deposited via HTTP).

`sender-name=CDATA`

This attribute may specify the textual name of the PPG (useful to human operators).

`reply-time=%Datetime`

This attribute specifies the date and time associated with the creation of the response. The time format is the same as that used in `deliver-before-timestamp`.

9.3.1 The progress-note Element

```
<!ELEMENT progress-note EMPTY >
<!ATTLIST progress-note
          stage                 CDATA             #REQUIRED
          note                  CDATA             #IMPLIED
          time                  %Datetime;        #IMPLIED
>
```

The `progress-note` element allows the PPG to specify the outcome of various stages of processing of the submitted `push-message`. There should be one `progress-note` element for each stage reported. The support of the `progress-note` element by the PPG is OPTIONAL. This element is useful for debugging implementations of Push Initiators.

Attributes

`stage=CDATA`

This attribute contains text or a code that indicates the stage of message processing completed. The actual value for this attribute is implementation specific.

`note=CDATA`

This attribute contains a textual description of the outcome of the stage completed.

`time=%Datetime`

This attribute specifies the date and time that the stage completed. The time format is the same as that used in `deliver-before-timestamp`.

9.3.2 The response-result Element

```
<!ELEMENT response-result EMPTY >
<!ATTLIST response-result
          code                    CDATA              #REQUIRED
          desc                    CDATA              #IMPLIED
>
```

The `response-result` element allows the PPG to specify a code for the outcome of the submission of the `push-message` operation. This code represents the immediate status after the message has been submitted. Another element (`resultnotification-response`) is used to indicate the final outcome. The detection of a response-code indicates to the Push Initiator that no more progress notes are to be expected for this message.

Attributes

`code=CDATA`

This attribute contains a code that indicates the status of the submission. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the outcome of the submission.

9.4 The cancel-message Element

```
<!ELEMENT cancel-message ( address* ) >
<!ATTLIST cancel-message
          push-id                 CDATA              #REQUIRED
>
```

The `cancel-message` element is used by the Push Initiator to cancel a message that was previously submitted. The push-id should be that of the message which is to be cancelled. If `address` is not specified, then all messages with the specified push-id are to be cancelled. Multiple addresses may be specified.

Direction: (Push Initiator→PPG)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the `push-message` that is to be cancelled.

9.5 The cancel-response Element

```
<!ELEMENT cancel-response ( cancel-result+ ) >
<!ATTLIST cancel-response
          push-id                 CDATA              #REQUIRED
>
```

The `cancel-response` element is used by the PPG to inform the Push Initiator of the outcome of a `cancel-message` that was previously submitted. The `push-id` can be used by the Push Initiator to associate the response with the related cancel-message.

Direction: (PPG→Push Initiator)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the push-message that is to be cancelled.

9.5.1 The cancel-result Element

```
<!ELEMENT cancel-result ( address* ) >
<!ATTLIST cancel-result
          code                  CDATA           #REQUIRED
          desc                  CDATA           #IMPLIED
>
```

The `cancel-result` element contains the result of the cancel operation. The `address` specifies which client the result pertains to. If no address is present, the result is for all addresses associated with the specified `push-id.`

The cancel function may not be supported on all wireless networks. Those PPGs that do not support this function MUST reply with the “`Not Implemented`” status code in `cancel-result`.

Direction: (PPG→Push Initiator)

<u>Attributes</u>

`code=CDATA`

This attribute contains a code that indicates the outcome of the cancel operation. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the outcome of the submission.

9.6 The resultnotification-message Element

```
<!ELEMENT resultnotification-message ( address, quality-of-service? ) >
<!ATTLIST resultnotification-message
          push-id               CDATA           #REQUIRED
          sender-address        CDATA           #IMPLIED
          sender-name           CDATA           #IMPLIED
          received-time         %Datetime;      #IMPLIED
          event-time            %Datetime;      #IMPLIED
          message-state         %State;          #REQUIRED
          code                  CDATA           #REQUIRED
          desc                  CDATA           #IMPLIED
>
```

The `resultnotification-message` element provides a means to specify the outcome of a submitted message for a specific recipient after the final result is known. This includes message delivery, expiration, cancellation, etc. The `quality-of-service` element specifies the delivery methods used if the message delivery was successful. It MUST be included if it was present in the push submission.

Direction: (PPG→Push Initiator)

<u>Attributes</u>

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the corresponding `push-message`. It can be used to match the result to the message.

`sender-address=CDATA`

This attribute contains the address of the PPG.

`sender-name=CDATA`

This attribute specifies the textual name of the PPG.

`received-time=%Datetime`

This attribute specifies the time at which the message was received at the PPG (from the Push Initiator). It may be used by the PI to calculate latency between the push submission time and the time at which the message reached its final disposition. The time format is the same as that used in `deliver-before-timestamp`.

`event-time=%Datetime`

This attribute specifies the time at which the message reached its final disposition. The time format is the same as that used in `deliver-before-timestamp`.

`message-state=%State`

This attribute indicates the state of the message. "`rejected`" indicates that the message was not accepted. "`pending`" indicates that the message is in process. "`delivered`" indicates that the message was successfully delivered to the client. "`undeliverable`" indicates that the message could not be delivered because of a problem. "`expired`" means that the message reached the maximum age allowed by PPG policy or could not be delivered by the time specified in the push submission. "`aborted`" indicates that the client aborted the message. "`timeout`" indicates that the delivery process timed out. "`cancelled`" indicates that the message was cancelled through the cancel operation. "`unknown`" indicates that the PPG does not know the state of the message.

`code=CDATA`

This attribute contains a code that indicates the final status of the message. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the outcome of the submission.

9.7 The resultnotification-response Element

```
<!ELEMENT resultnotification-response ( address ) >
<!ATTLIST resultnotification-response
          push-id              CDATA          #REQUIRED
          code                 CDATA          #REQUIRED
          desc                 CDATA          #IMPLIED
>
```

The `resultnotification-response` is sent by the Push Initiator to confirm receipt of the `resultnotification-message`.

Direction: (Push Initiator→PPG)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the corresponding `push-message`. It can be used to match the result to the message.

`code=CDATA`

This attribute contains a code that indicates the whether the `resultnotification-message` was received correctly or not. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the outcome of the submission.

9.8 The statusquery-message Element

```
<!ELEMENT statusquery-message ( address* ) >
<!ATTLIST statusquery-message
          push-id              CDATA          #REQUIRED
>
```

The `statusquery-message` element is used by the Push Initiator to check status of a message that was previously submitted. The `push-id` should be that of the message for which status is desired. If `address` is not specified, then status for all messages with the specified `push-id` is requested. Multiple addresses may be specified.

Direction: (Push Initiator→PPG)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the push-message for which status is desired.

9.9 The statusquery-response Element

```
<!ELEMENT statusquery-response ( statusquery-result+ ) >
<!ATTLIST statusquery-response
          push-id               CDATA            #REQUIRED
>
```

The `statusquery-response` element is used by the PPG to inform the Push Initiator of the status of a message that was previously submitted. The `push-id` can be used by the Push Initiator to associate with the related statusquery-message.

The status function may not be supported on all wireless networks. Those PPGs that do not support this function MUST reply with the "`Not Implemented`" status.

Direction: (PPG→Push Initiator)

Attributes

`push-id=CDATA`

This attribute is the message ID that was assigned by the Push Initiator for the push-message for which status is desired.

9.9.1 The statusquery-result Element

```
<!ELEMENT statusquery-result ( address*, quality-of-service? ) >
<!ATTLIST statusquery-result
          event-time            %Datetime;       #IMPLIED
          message-state         %State;          #REQUIRED
          code                  CDATA            #REQUIRED
          desc                  CDATA            #IMPLIED
>
```

The `statusquery-result` element contains the status of a message that was previously submitted. The `address` specifies which client the result pertains to. If no address is present, the result is for all addresses associated with the specified `push-id.` The `quality-of-service` element specifies the delivery methods used if the message delivery was successful. It MUST be included if it was present in the push submission.

Attributes

`event-time=%Datetime`

This attribute specifies the time at which the message reached its final disposition. The time format is the same as that used in `deliver-before-timestamp`.

`message-state=%State`

This attribute indicates the state of the message. The valid values are described in `resultnotification-message`.

`code=CDATA`

This attribute contains a code that indicates the status of the message. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the status of the submission.

9.10 The ccq-message Element

```
<!ELEMENT ccq-message ( address ) >
<!ATTLIST ccq-message
          query-id               CDATA             #IMPLIED
          app-id                 CDATA             #IMPLIED
>
```

The `ccq-message` element is used by the Push Initiator to request device capabilities for a specified device.

Direction: (Push Initiator→PPG)

Attributes

`query-id=CDATA`

The Push Initiator assigns the `query-id`. It is used to associate responses to `ccq-message`. The Push Initiator is responsible for its uniqueness internal to the Push Initiator.

`app-id=CDATA`

The `app-id` is the ID of the application [PushMsg] that the Push Initiator will target with a subsequent push message. The PPG may use the `app-id` attribute value to aid in selecting the subset of client capability information to return.

9.11 The ccq-response Element

```
<!ELEMENT ccq-response ( address ) >
<!ATTLIST ccq-response
          query-id               CDATA             #IMPLIED
          code                   CDATA             #REQUIRED
          desc                   CDATA             #IMPLIED
>
```

The `ccq-response` element is used by the PPG to inform the Push Initiator of the capabilities of a device. The `query-id` can be used by the Push Initiator to associate with the related `ccq-message`.

Direction: (PPG→Push Initiator)

Attributes

`query-id=CDATA`

The Push Initiator assigns the `query-id`. It is used to associate responses to `ccq-message`.

`code=CDATA`

This attribute contains a code that indicates the status of the query. See Status Codes (section 9.13).

`desc=CDATA`

This attribute contains a textual description of the outcome of the query.

9.12 The badmessage-response Element

```
<!ELEMENT badmessage-response EMPTY >
<!ATTLIST badmessage-response
```

```
          bad-message-fragment   CDATA            #REQUIRED
>
```

The `badmessage-response` element is used in response to messages that are unrecognisable. A fragment of the unrecognisable message should be contained in the `bad-message-fragment` attribute.

Direction: (PPG→Push Initiator)

Attributes

`bad-message-fragment=CDATA`

This attribute contains a fragment of the bad message.

9.13 Status Codes

The status code is a four digit numeric value. The first digit of the status code indicates the class of the code. There are 5 classes:

- 1xxx: Success - The action was successfully received, understood, and accepted.
- 2xxx: Client Error - The request contains bad syntax or cannot be fulfilled.
- 3xxx: Server Error -The server failed to fulfil an apparently valid request.
- 4xxx: Service Failure - The service could not be performed. The operation may be retried.
- 5xxx: Mobile Device Abort - The mobile device aborted the operation.

Status codes are extensible. The Push Initiator and the PPG MUST understand the class of a status code. Unrecognised codes MUST be treated as the x000 code for that class. For implementation specific codes, the numbers in the range x500-x999 should be used.

The table below lists the currently defined status codes and their meanings.

			response-result	**cancel-result**	**resultnotification-message**	**resultnotification-response**	**statusquery-result**	**ccq-response**
Code	**Description**							
1000	OK	The request succeeded.		x		x	x	x
1001	Accepted for Processing	The request has been accepted for processing.	x					
2000	Bad Request	Not understood due to malformed syntax.	x	x		x	x	x
2001	Forbidden	The request was refused.	x	x			x	x

			response-result	cancel-result	resultnotification-message	resultnotification-response	statusquery-result	ccq-response
Code	**Description**							
2002	Address Error	The client specified was not recognised.	x	x			x	x
2003	Address Not Found	The address specified was not found.		x			x	
2004	`Push ID` Not Found	The `push-idPush ID` specified was not found.		x			x	
2005	Capabilities Mismatch	The capabilities assumed by the PI were not acceptable for the client specified.	x		x			
2006	Required Capabilities Not Supported	The input is in a form not supported by the client.	x		x			
2007	Duplicate push-id	The push-id supplied is not unique within the PPG.	x					
3000	Internal Server Error	Server could not fulfil request due to internal error.	x	x			x	x
3001	Not Implemented	Server does not support the requested operation.		x			x	x
3002	Version not Supported	The server refuses to support the protocol version indicated.	x	x		x	x	x
3003	Not Possible	Action not possible because message is no longer available.		x			x	
3004	Capability Matching not Supported	The PPG does not support client capability information provided in a push message.	x					
3005	Multiple Addresses Not Supported	The PPG does not support an operation that specified multiple recipients.	x	x			x	
3006	Transformation Failure	The PPG was unable to perform a transformation on the message.						
3007	Specified Delivery Method Not	The PPG could not perform the	x		x		x	

Code	Description		response-result	cancel-result	resultnotification-message	resultnotification-response	statusquery-result	ccq-response
	Possible	confirmed or unconfirmed delivery specified.						
3008	Capabilities Not Available	Client capabilities for the specified client are not available.						x
3009	Required Network Not Available	The network requested is not available.	x		x		x	
3010	Required Bearer Not Available	The bearer requested is not available.	x		x		x	
4000	Service Failure	The service failed. The client may re-attempt the operation.			x		x	
4001	Service Unavailable	The server is busy.			x		x	
5xxx	Mobile Client Aborted	The mobile client aborted the operation.			x		x	

9.14 Status Code Definitions

Each status code is described below.

9.14.1 Success 1xxx

This class of code indicates that the message or response was successfully received, understood, and accepted.

9.14.1.1 OK 1000

This code indicates that the action requested was successful. It is used in response messages to say that the request was successfully carried out.

9.14.1.2 Accepted for Processing 1001

This code indicates that the request has been accepted for processing, but the final outcome is not yet known. This code is used in response to a push submission to indicate that the message has been received by the PPG and seems to be well formed and valid.

9.14.2 Client Error 2xxx

This class of code indicates that the request contains bad syntax or cannot be fulfilled.

9.14.2.1 Bad Request 2000

This code indicates that the syntax of the message was not understood.

9.14.2.2 Forbidden 2001

The request was refused.

9.14.2.3 Address Error 2002

The address supplied in the request was not in a recognised format or the PPG ascertained that the address was not valid for the network because it was determined not to be serviced by this PPG.

9.14.2.4 Address Not Found 2003

The address supplied in the request could not be located by the PPG. This code is returned when an operation is requested on a previously submitted message and the PPG can not find the message for the address specified.

9.14.2.5 Push-id Not Found 2004

The Push ID supplied in the request could not be located by the PPG. This code is returned when an operation is requested on a previously submitted message and the PPG can not find the message for the Push ID specified.

9.14.2.6 Capabilities Mismatch 2005

The PPG has determined that the capabilities supported by the mobile device do not match those supplied by the Push Initiator during the push submission. The PI has expected certain capabilities that are not available in the device.

9.14.2.7 Required Capabilities Not Supported 2006

The PPG has determined that the mobile device does not have the capabilities to support the message that was submitted by the PI.

9.14.2.8 Duplicate push-id 2007

The PPG has determined that the Push Initiator has violated the protocol rule that each new push submission must have a unique push ID.

9.14.3 Server Error 3xxx

This class of code indicates that the server failed to fulfil an apparently valid request.

9.14.3.1 Server Error 3000

The server failed to fulfil an apparently valid request.

9.14.3.2 Not Implemented 3001

The requested operation is not implemented in the PPG.

9.14.3.3 Version Not Supported 3002

The version of PAP contained in the request is not supported.

9.14.3.4 Not Possible 3003

The request could not be carried out because it is not possible. This code is normally used as a result of a cancel or status query on a message that is no longer available for cancel or status. The PPG has recognized the message in question, but it cannot fulfill the request because the message is already complete or status is no longer available.

9.14.3.5 Capability Matching Not Supported 3004

The PPG does not support the matching of client capabilities supplied by the push initiator in a push submission with those of the mobile device.

9.14.3.6 Multiple Addresses Not Supported 3005

The PPG does not support this operation on multiple recipients. The operation MAY be resubmitted as multiple single recipient operations.

9.14.3.7 Transformation Failure 3006

The PPG could not perform a transformation on the message.

9.14.3.8 Specified Delivery Method Not Possible 3007

The PPG could not deliver the message using the delivery-method specified in the request.

9.14.3.9 Capabilities Not Available 3008

The PPG does not know capabilities for the specified client. This code is typically in response to a capabilities query when the PPG cannot access the capabilities information for the mobile device.

9.14.3.10 Required Network Not Available 3009

The message could not be delivered using the network specified in the request.

9.14.3.11 Required Bearer Not Available 3010

The message could not be delivered using the bearer specified in the request.

9.14.4 Service Failure 4xxx

This class of code indicates that the service could not be performed. The operation may be retried.

9.14.4.1 Service Failure 4000

The service could not be performed. The operation may be retried.

9.14.4.2 Service Unavailable 4001

This code indicates that the server could not honour the request because the server is busy.

9.14.5 Mobile Client Abort 5xxx

The 5xxx codes are used to return three digit mobile client abort codes to the Push Initiator. The mobile client abort codes are placed into the 5xxx code as follows: 5abc where abc is a three digit code returned by the mobile client.

10. Version Control

PAP uses a “<major>.<minor>” numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further PAP communication. The <minor> number is incremented when the changes made to the protocol add features which do not change the general message structure other than adding optional attributes, or may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed in such a manner as to change the structure of an element or add new required attributes. Note that the major and minor numbers MUST be treated as separate integers and that each MAY be incremented higher than a single digit. Thus, PAP 2.4 is a lower version than PAP 2.10.

The PAP version number is placed in the public identifier [XML] of the DTD and will be changed with each revision to reflect the version number. Each new version of PAP will have its version number increased, even if the change does not affect the DTD (e.g. a semantic change). The file name of the DTD should also be changed so that it can easily be associated with the public identifier.

Example:

For version 1.0, the public identifier is: "-//WAPFORUM//DTD PAP 1.0//EN"

and the DTD filename is specified as: `"http://www.wapforum.org/DTD/pap_1.0.dtd"`

For version X.Y, the public identifier is: "-//WAPFORUM//DTD PAP X.Y//EN"

and the DTD filename is specified as: `"http://www.wapforum.org/DTD/pap_X.Y.dtd"`

Future versions of PAP should be backward compatible with older versions as much as possible. Element and attribute names should not be changed between versions. New elements and attributes should be ignored by older implementations when the minor number has been incremented.

11. Capabilities Negotiation

There are two methods a Push Initiator may use to determine the capabilities of the wireless device - either query before submission, or use a previously configured profile (e.g. configured via subscription). The Push Initiator may also pass the assumed capabilities to the PPG during a push submission. This allows the PPG to determine whether the submitted message is appropriate for the client.

11.1.1 Capabilities Query

The PAP makes a method available for determining device capabilities for push. The Push Initiator may query the PPG for information about a specific device's capabilities using the `ccq-message`.

11.1.2 Subscription

Typically push messages are desirable to the user of the wireless device and in fact the user often must subscribe to a service in order to receive these messages. During the subscription process the user of the device may communicate the device capabilities to the Push Initiator or other entity that may handle the subscription process for the Push Initiator. The Push Initiator can then use this information when submitting push messages for that subscriber. In this scenario, a capabilities query is not needed.

11.1.3 Assumed Capabilities

The Push Initiator MAY inform the PPG of the capabilities that have been assumed by the Push Initiator for a specific message during the submission of that message by specifying the capabilities in the third entity in the push message. The PPG MAY verify that the client has the capabilities desired and if some of the known capabilities are not sufficient, and the PPG is not prepared to perform possible transformations, then the PPG SHOULD inform the Push Initiator of the problem and abort the message.

For multiple recipient submissions, each recipient is handled as if it were an individual submission. Errors are reported using `resultnotification-message`, if requested by the Push Initiator.

12. PAP Reference Information

Push Access Protocol (PAP) is an application of [XML] version 1.0.

12.1 Document Identifiers

12.1.1 SGML Public Identifier

Editor's note: This identifier has not yet been registered with the IANA or ISO 9070 registrar

```
-//WAPFORUM//DTD PAP 1.0//EN
```

12.1.2 PAP Media Type

Textual form:

```
      application/xml
```

12.2 Document Type Definition (DTD)

```
<!--
Push Access Protocol (PAP) Document Type Definition.
PAP is an XML language. Typical usage:
   <?xml version="1.0"?>
   <!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 1.0//EN"
             "http://www.wapforum.org/DTD/pap_1.0.dtd">
   <pap>
      ...
   </pap>
-->

<!ENTITY % Datetime "CDATA">     <!-- ISO date and time -->
<!ENTITY % State  "(rejected | pending
                  | delivered | undeliverable
                  | expired | aborted
                  | timeout | cancelled | unknown)">
                                 <!-- PPG Message State -->


<!ELEMENT pap                    ( push-message
                                 | push-response
                                 | cancel-message
                                 | cancel-response
                                 | resultnotification-message
                                 | resultnotification-response
                                 | statusquery-message
                                 | statusquery-response
                                 | ccq-message
                                 | ccq-response
                                 | badmessage-response) >
<!ATTLIST pap
          product-name           CDATA             #IMPLIED
>



<!-- ========================================= -->
<!-- Declaration of push submission message    -->
<!-- ========================================= -->

<!--this message goes from the Push Initiator to the push proxy gateway-->

<!ELEMENT push-message ( address+, quality-of-service? ) >
<!ATTLIST push-message
          push-id                    CDATA             #REQUIRED
          deliver-before-timestamp   %Datetime;        #IMPLIED
          deliver-after-timestamp    %Datetime;        #IMPLIED
          source-reference           CDATA             #IMPLIED
          ppg-notify-requested-to    CDATA             #IMPLIED
          progress-notes-requested   ( true | false )  "false"
>

<!ELEMENT address EMPTY >
<!ATTLIST address
          address-value          CDATA             #REQUIRED
>

<!ELEMENT quality-of-service EMPTY >
<!ATTLIST quality-of-service
          priority               ( high | medium | low )        "medium"
          delivery-method        ( confirmed | preferconfirmed
                                 | unconfirmed | notspecified ) "notspecified"
          network                CDATA                          #IMPLIED
          network-required       ( true | false )               "false"
          bearer                 CDATA                          #IMPLIED
          bearer-required        ( true | false )               "false"
>



<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT push-response ( progress-note*, response-result ) >
```

```
<!ATTLIST push-response
          push-id                CDATA            #REQUIRED
          sender-address         CDATA            #IMPLIED
          sender-name            CDATA            #IMPLIED
          reply-time             %Datetime;       #IMPLIED
>

<!ELEMENT progress-note EMPTY >
<!ATTLIST progress-note
          stage                  CDATA            #REQUIRED
          note                   CDATA            #IMPLIED
          time                   %Datetime;       #IMPLIED
>

<!ELEMENT response-result EMPTY >
<!ATTLIST response-result
          code                   CDATA            #REQUIRED
          desc                   CDATA            #IMPLIED
>


<!-- ========================================= -->
<!-- Declaration of cancel operation           -->
<!-- ========================================= -->

<!--this message goes from the Push Initiator to the push proxy gateway-->

<!ELEMENT cancel-message ( address* ) >
<!ATTLIST cancel-message
          push-id                CDATA            #REQUIRED
>

<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT cancel-response ( cancel-result+ ) >
<!ATTLIST cancel-response
          push-id                CDATA            #REQUIRED
>

<!ELEMENT cancel-result ( address* ) >
<!ATTLIST cancel-result
          code                   CDATA            #REQUIRED
          desc                   CDATA            #IMPLIED
>
```

```
<!-- ========================================= -->
<!-- Declaration of notify result operation    -->
<!-- ========================================= -->

<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT resultnotification-message ( address, quality-of-service? ) >
<!ATTLIST resultnotification-message
          push-id               CDATA            #REQUIRED
          sender-address        CDATA            #IMPLIED
          sender-name           CDATA            #IMPLIED
          received-time         %Datetime;       #IMPLIED
          event-time            %Datetime;       #IMPLIED
          message-state         %State;          #REQUIRED
          code                  CDATA            #REQUIRED
          desc                  CDATA            #IMPLIED
>

<!--this message goes from the Push Initiator to the push proxy gateway-->

<!ELEMENT resultnotification-response ( address ) >
<!ATTLIST resultnotification-response
          push-id               CDATA            #REQUIRED
          code                  CDATA            #REQUIRED
          desc                  CDATA            #IMPLIED
>


<!-- ========================================= -->
<!-- Declaration of statusquery operation      -->
<!-- ========================================= -->

<!--this message goes from the Push Initiator to the push proxy gateway-->

<!ELEMENT statusquery-message ( address* ) >
<!ATTLIST statusquery-message
          push-id               CDATA            #REQUIRED
>

<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT statusquery-response ( statusquery-result+ ) >
<!ATTLIST statusquery-response
          push-id               CDATA            #REQUIRED
>
```

```
<!ELEMENT statusquery-result ( address*, quality-of-service? ) >
<!ATTLIST statusquery-result
          event-time              %Datetime;        #IMPLIED
          message-state           %State;           #REQUIRED
          code                    CDATA             #REQUIRED
          desc                    CDATA             #IMPLIED
>


<!-- ============================================== -->
<!-- Declaration of capabilities query operation    -->
<!-- ============================================== -->

<!--this message goes from the Push Initiator to the push proxy gateway-->

<!ELEMENT ccq-message ( address ) >
<!ATTLIST ccq-message
          query-id                CDATA             #IMPLIED
          app-id                  CDATA             #IMPLIED
>

<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT ccq-response ( address ) >
<!ATTLIST ccq-response
          query-id                CDATA             #IMPLIED
          code                    CDATA             #REQUIRED
          desc                    CDATA             #IMPLIED
>
<!-- ============================================== -->
<!-- Declaration of bad message response message    -->
<!-- ============================================== -->


<!--this message goes from the push proxy gateway to the Push Initiator-->

<!ELEMENT badmessage-response EMPTY >
<!ATTLIST badmessage-response
          bad-message-fragment   CDATA             #REQUIRED
>
```

13. Examples

13.1 push-message Example

Below is an example of PAP push-message with a WML deck in a MIME multipart using text/vnd.wap.wml. In this example, the optional capabilities expected entity has been provided by the PI. The WML in this example is not very useful, but serves as a simple example.

```
Content-Type: multipart/related; boundary=asdlfkjiurwghasf;
                                 type="application/xml"

--asdlfkjiurwghasf
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>
  <push-message push-id="9fjeo39jf084@pi.com">
    <address address-value="wappush=12345/type=user@ppg.operator.com"></address>
  </push-message>
</pap>

--asdlfkjiurwghasf
Content-Type: text/vnd.wap.wml
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                     "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card>
    <p>Hello World!</p>
  </card>
</wml>

 --asdlfkjiurwghasf
 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/UAPROF/ccppschema1.0#">
         <!--WAP Browser vendor site: Default description of WAP properties -->
  <rdf:Description>
    <prf:WapVersion>1.1</prf:WapVersion>
    <prf:WmlDeckSize>1400 octets</prf:WmlDeckSize>
    <prf:WapDeviceClass>A </prf:WapDeviceClass>
    <prf:WapPushMsgSize>1400 octets</prf:WapPushMsgSize>
    <prf:WmlVersion>
      <rdf:Bag>
        <rdf:li>1.1</rdf:li>
           </rdf:Bag>
        </prf:WmlVersion>
  </rdf:Description>
</rdf:RDF>

--asdlfkjiurwghasf--
```

14. Push Access Protocol over HTTP

This section describes how the Push Access Protocol is transported using HTTP [RFC2616].

The HTTP POST method and response are used to transport the Push Access Protocol.

Using HTTP, each operation is begun with an HTTP POST method containing the information for delivery to the PPG or Push Initiator. Upon receipt of the POST, the receiving server (PPG or Push Initiator) replies with an HTTP response containing the response for the operation.

For added security, HTTP may be used with SSL or TLS.

14.1 Addressing

The push proxy gateway URL specifies the application on the PPG that handles the delivery of WAP push data to a device.

Example HTTP POST addressed to an example WAP push path on a network named wireless_network:

```
POST /cgi-bin/wap_push.cgi HTTP/1.1
Host: www.wireless-network.com
………
```

14.2 Message Format

The message format is described in section 8. The HTTP message body transports the message. HTTP version 1.1 and later versions will be used.

14.3 Example

The following is an example of a multipart/related MIME body containing an XML document and a content entity, transported by the HTTP POST method.

```
POST  /cgi-bin/wap_push.cgi HTTP/1.1
Host: www.wireless-network.com
Date: Sun, 16 May 1999 18:13:23 GMT
Content-Type:           multipart/related;           boundary=asdlfkjiurwghasf;
type="application/xml"
Content-Length: 353

--asdlfkjiurwghasf
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>
  ..control..
</pap>

--asdlfkjiurwghasf
Content-Type: text/vnd.wap.wml
```

```
<?xml version="1.0"?>
      <!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card>
    <p>Hello world!</p>
  </card>
</wml>
--asdlfkjiurwghasf--
```

The wap_push.cgi process is the application which handles push messages for the wireless network. The XML document contains all control information needed by the PPG to deliver the message to the wireless client. The information intended for the wireless client is carried in an entity in a push message [PushMsg].

14.4 Message Responses

This section attempts to clarify what is expected in a response message in HTTP.

14.4.1 HTTP Response Codes

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.

15. Static Conformance Requirements

This static conformance clause defines a minimum set of features that should be implemented to support the Push Access Protocol. A feature can be optional (O), mandatory (M) or conditional (C (<condition>)). If optional/conditional features have labels (O.<n> or C.<n>), support of at least one in the group of options labelled by the same number is required.

15.1 Push Proxy Gateway Features

15.1.1 Validation

Item	Functionality	Reference	Status
PAP_VAL_001	Validate XML in control entity in push submission	6.1	M
PAP_VAL_002	Validate content entity	6.1	O
PAP_VAL_003	Validate addresses	6.1	O

15.1.2 Operations

Item	Functionality	Reference	Status
PAP_OPS_001	Push Submission	6.1	M
PAP_OPS_002	Result Notification	6.2	M
PAP_OPS_003	Push Cancellation	6.3	O
PAP_OPS_004	Status Query	6.4	O
PAP_OPS_005	Client Capabilities Query	6.5	O

15.1.3 Semantics

Item	Functionality	Reference	Status
PAP_SEM_001	Support of multiple addresses in messages	7.1	O
PAP_SEM_002	Support of multiple addresses in responses	7.1	O
PAP_SEM_003	Deliver after timestamp	9.2	O
PAP_SEM_004	Deliver before timestamp	9.2	O
PAP_SEM_005	Fail requests when QOS cannot be honoured.	9.2.2	M
PAP_SEM_006	Delivery-method in QOS	9.2.2	M
PAP_SEM_007	Priority delivery	9.2	O
PAP_SEM_008	Report progress notes	9.3	O
PAP_SEM_009	Support capabilities entity in push message	6.1	O