Specification

Push Request Protocol Specification

Document number: ST17455122

Revision: B

Contents

Document number: ST17455122
Revision: B

Brief description

This document describes the communication protocol between the WIG server and the Push client. The Push client is defined as the program that generates a push request to be sent to a mobile station. The HTTP protocol will be used on top of TCP/IP and the message will be a XML script that contains the WML document.

The XML script will be based on the style sheet proposed in WAP Push Access Protocol, see reference [2]. The Push Access Protocol is a proposed standard by WAP forum. The protocol between the Push client and the WIG server will support a subset of the features described in the WAP Push Access Protocol. This document will describe the features supported in this version of the WIG server.

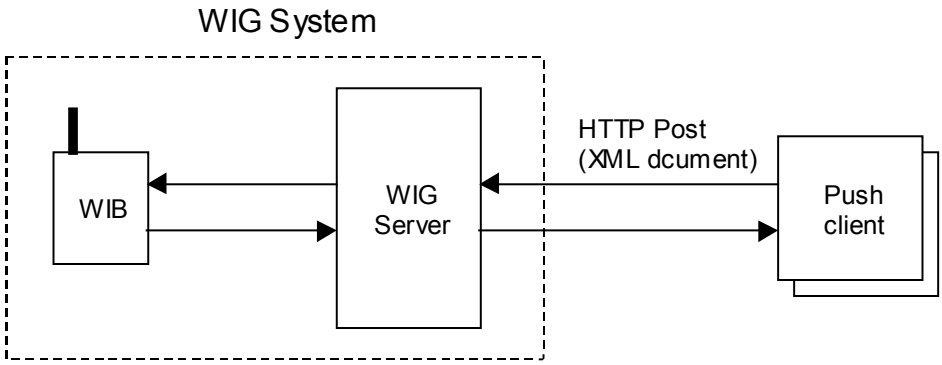

Figure 1 Pusch client and WIG system

Figure 1 above shows the basic system configuration for the WIG and the Push client.

Figure 2 below shows a scenario for how the Push client may interact with the WIG server and the Web server residing at the content provider.

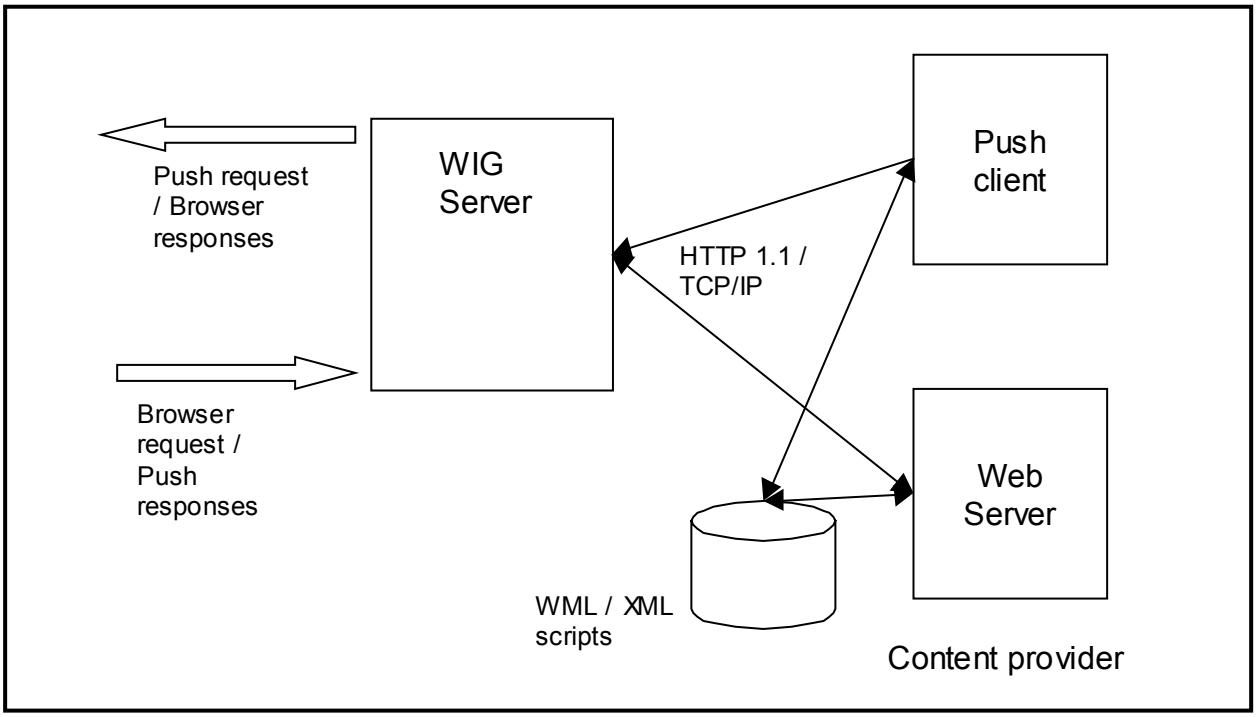

Figure 2 Overview of Push client and WIG server

The main difference between the proposed WAP Push architecture and the Push architecture implemented in the WIG server is that responses on Push requests will be handled as browser requests in the WIG server. The WAP Push architecture defines the Push responses as XML messages that shall be returned to the Push client.

References

Ref.	Title	Document No.
[1]	Hypertext Transfer Protocol HTTP/1.1, RFC 2068, Version 1997-01, http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html	
[2]	WAP Push Access Protocol – Proposed version 1608, November 1999.	WAP specification document.
[3]	WAP Push Architectural Overview – Proposed version 1608, November 1999.	WAP specification document.

Push client – WIG system protocol description

The protocols between the Push client and the WIG system are the standard Internet protocols HTTP and TCP/IP. Only the HTTP Post method of the HTTP protocol is supported, see reference [1].

The HTTP protocol is a request/response protocol, see Appendix. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

On top of the HTTP protocol the WAP Push Access Protocol (PAP) is used. The PAP defines the content of the document sent using the HTTP protocol.

The PAP defines that the document will consist of three parts, a control section, a message section and a client capabilities section. The PAP is based on XML and uses a data type definition (DTD) document, pap_1.0.dtd, specified by the WAP Forum. The DTD defines all allowed fields in the control section and in the capabilities section. The message section consist of a valid WML document.

To handle secure push transaction the secure socket layer (SSL) may be used between the WIG system and the push client.

Request message

The WIG system supports HTTP request messages from the push client including the phone number to the WIB, a confirmation flag and a WML document, see reference [3]. The confirmation flag indicates whether the push client expects a confirmation of delivery to the WIB.

A general HTTP request message

A general HTTP request message has the following structure

```
Request =   Request-Line
            * ( general-header
              | request-header
              | entity-header )
            CRLF
            [ message-body ]
```

where

```
Request-Line = Method SP Request-URI SP HTTP-Version CRLF
```

Push client request

The WIG system supports the following HTTP request message settings specifying an push client request to the WIG system.

Request-Line

```
Method = "POST"
Request-URI = "/SendToWIGClient.asp"
HTTP-Version = "HTTP/1.1"
```

Observe that the value for the Request-URI is just an example. The Request-URI is currently not used by the WIG system.

Request-header

```
request-header = Host
```

The Host request-header field specifies the Internet host and port number of the WIG Server.

```
Host = "Host" ":" host [ ":" port ]

host = <A legal Internet host domain name or IP address (in
dotted-decimal form)>

port = *DIGIT
```

If the port is empty or not given, port 80 is assumed.

Entity-header

```
entity-header = Content-Type | Content-Length
```

The Content-Type entity-header field indicates the media type of the message-body.

```
Content-Type = "Content-Type" ":" media-type

media-type = "application/x-www-form-urlencoded"
```

The Content-Length entity-header field indicates the size of the message-body in decimal number of octets.

```
Content-Length = "Content-Length" ":" 1*DIGIT
```

Document-body

The document-body of the HTTP request contains the following:

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

--<boundary>
Control section

--<boundary>
Message section (the WML script)

--<boundary>
Client capabilities

--<boundary>

An example of a message may look like:

```
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=”+45700000000”>
     </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—
```

The control section is used to define the following data (among other things):

- MSISDN - The phone number to a WIB.
- Confirmation flag and the URL where the confirmation shall be sent.
- A unique ID for the push message.

See reference [3] for a full description of the controll and client capabilities section.

Example

Example of an URL encoded HTTP request message:

```
POST /SendToWIB.asp HTTP/1.1<CR><LF>
Host: sim_cps2:123<CR><LF>
Content-Type: application/x-www-form-
urlencoded<CR><LF>
Content-Length: 436<CR><LF>
<CR><LF>
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@content-provider.com"
     deliver-before-timestamp="1999-11-19T17:00:00Z"
     deliver-after-timestamp="1999-11-19T12:00:00Z"
     ppg-notify-requested-to="www.content-
provider.com">
     <address
          address-value="+45700000000">
          <quality-of-service
               priority="high">
               delivery-method="confirmed">
          </quality-of-service>
     </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
```

Response message

The WIG system supports two types of response messages to the push client: HTTP acknowledge and confirmation of delivery.

The confirm message contains the control section of the push request. The control section have the MSISDN number in the address field and typically also unique identifier for the push request.

If the push message requires a response the WML document must contain a “GO HREF” element that indicates where the response shall be sent.

A general HTTP response message

A general HTTP response message has the following structure

```
Response =  Status-Line
            * ( general-header
              | response-header
              | entity-header )
            CRLF
            [ message-body ]
```

where

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

WIG system OK acknowledge

The WIG system supports the following HTTP acknowledge message specifying that the push message has been received by the WIG system.

Status-Line

```
HTTP-Version = "HTTP/1.1"
Status-code = "200"     Received push message OK
Reason-Phrase = <TEXT, excluding CR, LF>
```

WIG system error acknowledge

The WIG system supports the following HTTP acknowledge message settings specifying an error response from WIG system when receiving the push message.

Status-Line

```
HTTP-Version = "HTTP/1.1"
Status-code = "500"       ; Internal Server Error
Reason-Phrase = <TEXT, excluding CR, LF>
```

Examples

Examples of URL encoded HTTP response messages:

WIB response

```
HTTP/1.1 200 OK-Content<CR><LF>
Server: WIG 2.0.0<CR><LF>
Connection: close<CR><LF>
Date: Mon, 18 Sep 2000 12:54:51 GMT<CR><LF>
Content-length: 32<CR><LF><CR><LF>
<WML><CR><LF>
Push message OK<CR><LF>
</WML><CR><LF>
```

Time out response

```
HTTP/1.1 408 Request Time-out<CR><LF>
Server: WIG 2.0.0<CR><LF>
Connection: close<CR><LF>
Date: Mon, 18 Sep 2000 12:54:51 GMT<CR><LF>
Content-length: 32<CR><LF><CR><LF>
<WML><CR><LF>
Time-out receiving push message<CR><LF>
</WML><CR><LF>
```

Error response

```
HTTP/1.1 500 Internal Server Error<CR><LF>
Server: WIG 2.0.0<CR><LF>
Connection: close<CR><LF>
Date: Mon, 18 Sep 2000 12:54:51 GMT<CR><LF>
Content-length: 32<CR><LF><CR><LF>
<WML><CR><LF>
WIG server error<CR><LF>
</WML><CR><LF>
```

WIG system confirm

The WIG system supports the following HTTP response message settings specifying a response from the WIB.

Status-Line

```
Method = "POST"
Response-URI = "/SendToWIGClient.asp"
HTTP-Version = "HTTP/1.1"
```

Observe that the value for the Response-URI is just an example. The Response-URI is currently not used by the WIG system.

entity-header

```
entity-header = Content-Type | Content-Length
```

The Content-Type entity-header field indicates the media type of the message-body.

```
Content-Type = "Content-Type" ":" media-type

media-type = "application/x-www-form-urlencoded"
```

The Content-Length entity-header field indicates the size of the message-body in decimal number of octets.

```
Content-Length = "Content-Length" ":" 1*DIGIT
```

message-body

The document-body of the HTTP request contains the following:

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

--<boundary>
Control section

--<boundary>

An example of a confirm message may look like:

```
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=”+45700000000”>
     </address>
</push-message>
</pap>

--asdlfkjiurwghasf
```

Examples

Example of an URL encoded HTTP confirm message:

```
POST /SendToWIB.asp HTTP/1.1<CR><LF>
Host: sim_cps2:123<CR><LF>
Content-Type: application/x-www-form-
urlencoded<CR><LF>
Content-Length: 436<CR><LF>
<CR><LF>
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@content-provider.com"
     ppg-notify-requested-to=”www.content-
provider.com”>
     <address
          address-value=”+45700000000”>
          <quality-of-service
               priority="high">
               delivery-method="confirmed">
          </quality-of-service>
     </address>
</push-message>
</pap>

--asdlfkjiurwghasf
```

WAP Push protocol overview

In the WAP technology a push operation occurs when a Push Initiator (Push client) transmits content to a client using either the Push Over-The-Air (OTA) protocol or the Push Access protocol. The normal situation is that the Push client is on the Internet and the WAP client is in the WAP domain and therefore a translating gateway is needed. This architecture is the same used with the WIG server. The Push Access protocol then defines the communication protocol between a Push client on the Internet and the gateway (WIG server).

The Push Access protocol supports the following operations:

- Push submission (Push client to WIG server)
- Result notification (WIG server to Push client)
- Push cancellation (Push client to WIG server)
- Status query (Push client to WIG server)
- Client capability query (Push client to WIG server)

All operations above will be described in this document but for WIG server 2.0 only the two first operations will be supported. The Push submission operation will also be restricted to only support submission to one single recipient i.e. there will be no support for multi- and broadcast messages in WIG server 2.0.

The WIG server 2.0 will be implemented in such a way that it will be easy to incorporate support for multicast, broadcast and the other operations in the future.

Addressing

There are three addresses involved in the Push Access protocol:

- WIG server address
- Mobile station address
- Result notification address

The WIG server address will be an IP address and port number. The Mobile station address is included as a part of the message content (XML tagged content). The Result notification address is given in the “ppg-notify-requested to” field.

Message format

A message will contain three parts, the control entity, the content entity and the capability entity. The “Content-Type” tag in the HTTP header shall be used to indicate the three parts. The structure if the message is:

HTTP header

```
POST <URI> HTTP 1.1\r\n
Accept: text/*\r\n
Accept-Charset: iso-8859-1;UTF-8\r\n
Accept-Encoding: identity\r\n
Accept-Language: \r\n
User-Agent: WIG Browser/<browser version>\r\n
Connection: close\r\n
Host: <host name>\r\n
Content-Type: multipart/related;boundary=asdlfkjiurwgh;
              type="application/xml\r\n
\r\n
```

--<boundary>

Control section

--<boundary>

Message section (the WML script)

--<boundary>

Client capabilities

--<boundary>

An example of a message may look like:

```
POST /helloworld.wml HTTP 1.1\r\n
Accept: text/*\r\n
Accept-Charset: iso-8859-1;UTF-8\r\n
Accept-Encoding: identity\r\n
Accept-Language: \r\n
User-Agent: WIG Browser/1.1\r\n
Connection: close\r\n
Host: sim_cps2\r\n
Content-Type: multipart/related;
boundary=asdlfkjiurwghasf;
type="application/xml"\r\n
\r\n
--asdlfkjiurwgh

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=”+45700000000”>
```

```
     </address>
</push-message>
</pap>

--asdlfkjiurwgh

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>

--asdlfkjiurwgh

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>

--asdlfkjiurwgh
```

The client capabilities part is optional and will not be supported in WIG 2.0. Also multiple addresses are allowed but that feature will not be supported in WIG 2.0.

Supported tags in the control section

The table below describes the supported tags in the control section. The remaining tags as defined by WAP Forum (reference [2]) are not supported.

Element	Attribute	Required/ Optional	Comment
push-message		R	The first element tag in a push message
	push-id	R	ID of push message given by push client.
	ppg-notify-requested-to	O	The address for the push notification/response.
Address		R	The address element
	address-value	R	The address to the recipient i.e. MSISDN
Quality-of-service		R	Element that is used for defining quality of service.
	delivery-method	O	Valid values are “confirmed”, “unconfirmed”, “notspecified”. This field tells if the Push client shall be confirmed. The value “preconfirmed” is not supported.

No response messages from the WIG server to the Push client will be supported since the responses on a push will be handled as a browser request. (Confirm, confirms to the push client that the message reached the MS)

Example of a Push request message

```
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@content-provider.com"
     ppg-notify-requested-to=”www.content-
provider.com”>
     <address
          address-value="123456789">
          <quality-of-service
               delivery-method="confirmed">
          </quality-of-service>
     </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
```

Data formats used

This section describes different data formats and recommendations of data formats to use for certain fields.

Date and time

Date and time values shall be given in the format:

YYYY-MM-DDThh:mm:ssZ

where YYYY is year, MM month, DD day, hh hour (00-23), mm minutes and ss seconds. Exampel of a valid date time string is:

1999-11-18T16:30:00Z

Push-ID field

The push-id is given by the push client and shall preferable be unique for each push message. A recommendation is to use the format

<number>@content-provider.com

where the last part comes from the email address to the content provider.

Appendix A: HTTP prototocol

The HTTP protocol is a request/response protocol, see reference [1]. A short summary follows.

```
HTTP-message = Request | Response
```

Both types of message consist of a start-line, one or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and an optional message-body.

```
Generic-message = start-line
                  *message-header
                  CRLF
                  [ message-body ]


start-line = Request-Line | Status-Line
```

Each message-header (header field) consists of a name followed by a colon (":") and the field value. Field names are not case sensitive. The field value may be preceded by any amount of LWS, though a single SP is preferred.

```
message-header = field-name ":" [ field-value ] CRLF

field-name = token
field-value = *( field-content | LWS )

field-content = <the OCTETs making up the field-value and
consisting of either *TEXT or combinations of token, tspecials,
and quoted-string>
```

Appendix B: Document Type Definition

```
<!--
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>