WAP Push Architectural Overview

Proposed Version 16-August-1999

Wireless Application Protocol
Push Architectural Overview

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"* [WAPArch].

This document outlines the WAP Push specifications, which together specify a service to push content to mobile devices via the WAP architecture. Please note that this document is informative, not normative: if there appears to be a discrepancy between this overview and a specification, follow the specification.

2. Document Status

This document is available online in the following formats:

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

This document is in DRAFT mode and may not be disclosed to non-members of the WAP Forum.

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

None; this is an informative document.

3.2 Informative references

[ISO8601] "Data elements and interchange formats - Information interchange - Representation of dates and times", International Organization For Standardization (ISO), 15-June-1988; and

"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

[HTTP] "Hypertext Transport Protocol – HTTP/1.1", R. Fielding et al., June 1999.
URI: http://www.ietf.org/rfc/rfc2616.txt

[PushOTA] "WAP Push OTA Specification", WAP Forum, 16-August-1999. URI: http://www.wapforum.org/

[PushPAP] "WAP Push Access Protocol Specification", WAP Forum, 16-August-1999.
URI: http://www.wapforum.org/

[PushPPG] "WAP Push Proxy Gateway Specification", WAP Forum, 16-August-1999.
URI: http://www.wapforum.org/

[RFC821] "Simple Mail Transfer Protocol", J. Postel, 1982-08-01. URI: http://www.ietf.org/rfc/rfc0821.txt

[RFC822] "Standard for the format of ARPA Internet text messages", D. Crocker, 1982-08-01.
URI: http://www.ietf.org/rfc/rfc0822.txt

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

[RFC2387] "The MIME Multipart/related content type", E. Levinson, August 1998.
URI: http://www.ietf.org/rfc/rfc2387.txt

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

[SI] "WAP Service Indication Specification", WAP Forum, 16-August-1999.
URI: http://www.wapforum.org/

[UAPROF] "WAP User Agent Profile", WAP Forum, 22-June-1999, URI: http://www.wapforum.org/

[WAPArch] "Wireless Application Protocol Architecture Specification", WAP Forum, 30-April-1998.
URI: http://www.wapforum.org/

[WBXML] "Binary XML Content Format Specification", WAP Forum, 16-June-1999.
URI: http://www.wapforum.org/

[WINA] "WAP Interim Naming Authority", WAP Forum,
URL:http://www.wapforum.org/wina/

[WML] "Wireless Markup Language Specification", WAP Forum, 16-June-1999.
URI: http://www.wapforum.org/

[WSP] "Wireless Session Protocol Specification", WAP Forum, Ltd., 1998-05-28.
URI: http://www.wapforum.org/

[WTLS] "Wireless Transport Layer Security Specification", WAP Forum, 11-February-1999.
URI: http://www.wapforum.org/

[XML] "Extensible Markup Language (XML)", W3C Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998. URI: http://www.w3.org/TR/REC-xml

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

The WAP Push framework introduces a means within the WAP effort to transmit information to a device without a previous user action. In the normal client/server model, a *client* requests a service or information from a *server*, which then responds in transmitting information to the client. This is known as “pull” technology: the client “pulls” information from the server. The World Wide Web is a typical example of pull technology, where a user enters a URL (the request) which is sent to a server, and the server answers by sending a Web page (the response) to the user.

In contrast to this, there is also “push” technology, which is also based on the client/server model, but where there is no explicit request from the client before the server transmits its content.

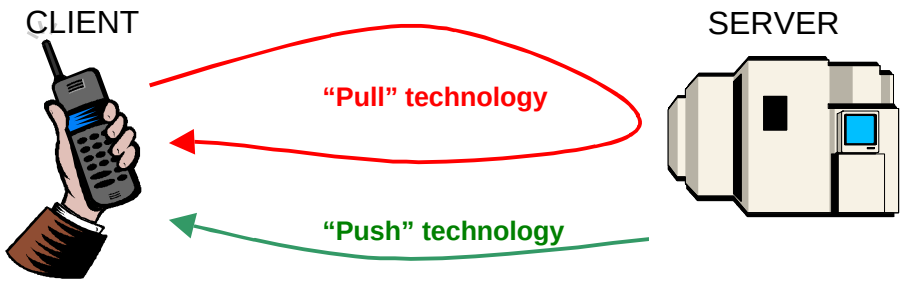

Figure 1 - Comparison of pull vs. push technology

Another way to say this is that whereas “pull” transactions of information are always initiated from the client, “push” transactions are server-initiated.

6. The Push Framework

A push operation in WAP occurs when a *Push Initiator* transmits *content* to a *client* using either the *Push Over-The-Air protocol* or the *Push Access Protocol*. In its simplest form, the architecture would look like this;

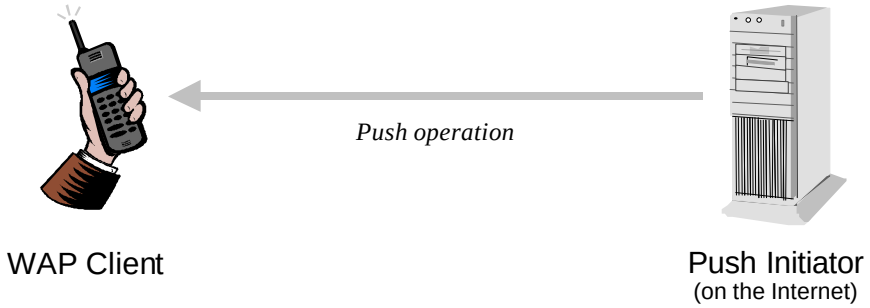

Figure 2 - The Push Framework at its simplest

However, the Push Initiator shares no protocol with the WAP Client: the Push Initiator is on the Internet, and the WAP Client is in the WAP domain. Therefore, the Push Initiator cannot contact the WAP Client without an intermediary, so we need to insert a translating gateway:

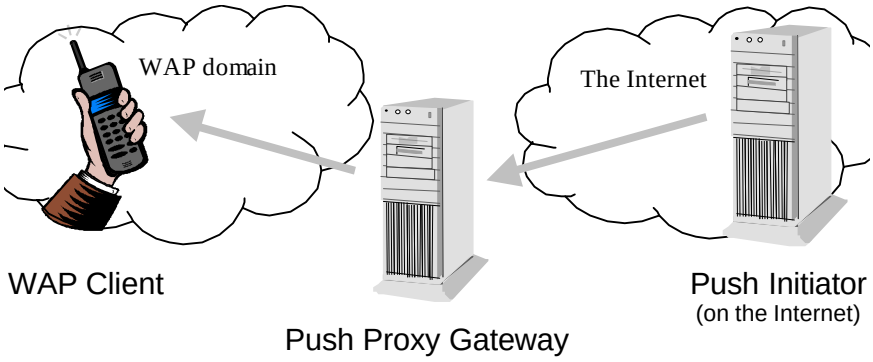

Figure 3 - The Push Framework with the Push Proxy Gateway

Thus, the Push Initiator contacts the Push Proxy Gateway (PPG) from the Internet side, delivering content for the destination client using Internet protocols. The PPG does what is necessary to forward the pushed content to the WAP domain, and the content is then transmitted over-the-air in the mobile network to the destination client.

In addition to providing simple proxy gateway services, the PPG may is capable of notifying the Push Initiator about the final outcome of the push operation, and it may wait for the client to accept or reject the content in two-way mobile networks. It may also provide the Push Initiator with client capability lookup services, letting a Push Initiator select the optimal flavour of this particular content for this particular client.

The Internet-side PPG access protocol is called the *Push Access Protocol*. The WAP-side (OTA) protocol is called the *Push Over-The-Air Protocol*. Thus, a revised schematic looks something like this:

The Push Access Protocol, or PAP, uses XML messages that may be tunnelled through various well-known Internet protocols, for example HTTP. The OTA protocol is based on WSP services (see [WSP]).

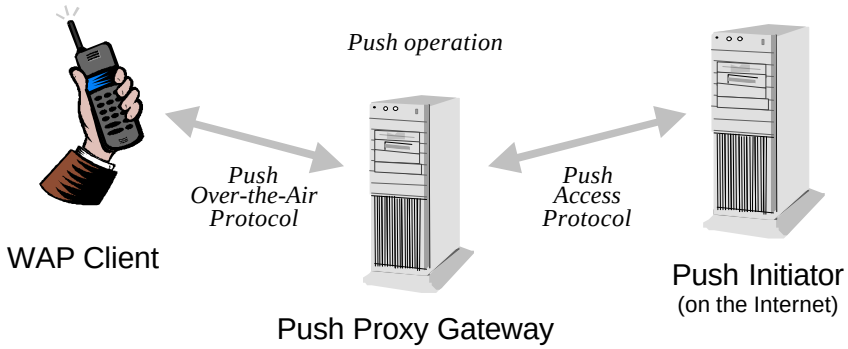

Figure 4 - The Push Framework with protocols

All of these entities are discussed in more detail in the following sections.

7. The Push Proxy Gateway

The Push Proxy Gateway (*PPG*) is the entity that does most of the work in the Push architecture. Its responsibilities include acting as an access point for content pushes from the Internet to the mobile network, and everything associated therewith (authentication, security, client control, etc).

As the PPG is the entry point to a mobile network, it is the owner of this gateway that decides the policies about who is able to gain access to the WAP network, who is able to push content and who is not and under which circumstances and parameters, etc.

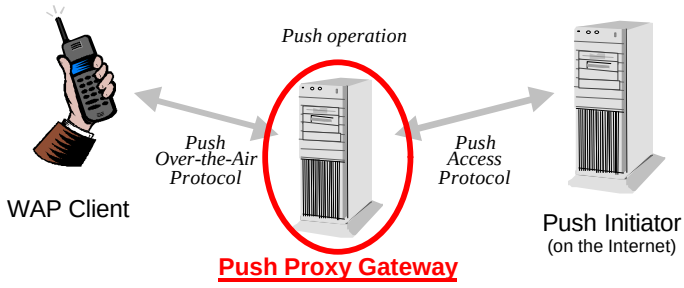

Figure 5 - PPG highlighted

It should be noted that the PPG functionality may be built into the (pull) WAP gateway defined in [WAP]; this would give the benefit of shared resources and shared sessions over-the-air.

7.1 Services Overview

The PPG hosts the Push framework with several services. First and foremost, it is the entry point for pushed content from the Internet destined for the WAP domain; this means it may be able to perform at least the following;

- Push initiator identification and authentication; access control
- Parsing of and error detection in content control information
- Client discovery services
- Address resolution
- Binary encoding and compilation of certain content types to improve efficiency OTA
- Protocol conversion

7.2 Access from the Internet Side

The PPG accepts pushed content from the Internet using the Push Access Protocol (see section 8). This content is divided into several sections using a multipart/related content type, where the first part contains information for the PPG itself. Such information includes recipient information, timeouts, callback requests, and similar pieces of information.

The PPG will acknowledge successful (or report unsuccessful) parsing of this control information, and may in addition report debug information about the content itself. It may also do a callback to the pushing server when the final status of the push submission (delivered, cancelled, expired, etc.) has been reached, if the push initiator so requests.

7.3 Message Handling Service

Once the content has been accepted for delivery, the PPG attempts to find the correct destination device and deliver the content to that client using the Push Over-The-Air protocol (see section 9). The PPG will attempt to deliver the content until a timeout expires; this timeout may be set by the push initiator and/or policies of the mobile operator.

The net result of this function is asynchronous operation from the Initiator's point of view (the Initiator need not wait on-line for the PPG to complete its delivery).

7.4 Encoding and Compilation

The PPG may encode WAP content types (e.g. WML and SI) into their binary counterparts, if feasible. This textual-to-binary translation would take place before delivery over-the-air. Other content types, which may be application-specific or just plain arbitrary, may be forwarded as received.

It is also possible for the Push Initiator to precompile its content into binary form, to take workload off the PPG (or for other reasons). When the PPG receives precompiled WML, WMLScript, or SIs, they are forwarded as received.

7.5 Multicast, Broadcast and Aliasing Considerations

As a side note, the PPG may implement addressing aliasing schemes to enable special multi- and broadcast cases, where special addresses may translate to a broadcast operation. This is an implementation detail and left to the discretion of the PPG implementer.

7.6 Client Capabilities Query Service

A Push Initiator may query the PPG for client capabilities and preferences, to create better-formatted content for a particular WAP device.

7.7 Reference

For more information, see [PushPPG].

8. The Push Access Protocol

The Push Access Protocol (*PAP*) is the means by which an Internet-based Push Initiator pushes content to a mobile network, addressing its PPG.

Care has been taken to ensure that this protocol can be tunnelled through any other or future Internet protocol in common use, although HTTP is chosen as the initially supported carrier.

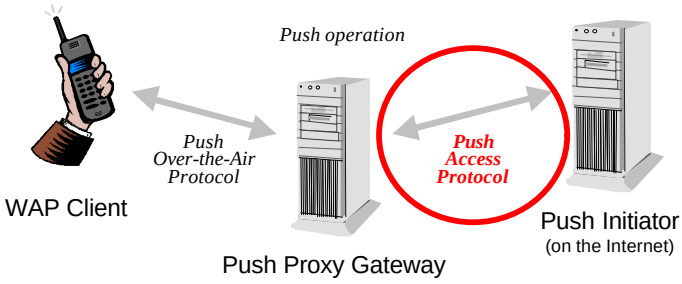

Figure 6 - PAP highlighted

8.1 General Structure

The PAP carries an XML-style entity (see [XML]) that may be bundled with other components in a multipart/related document (see [RFC2387]).

8.2 PAP Operations

The PAP supports the following operations:

- Push Submission (Initiator to PPG)
- Result Notification (PPG to Initiator)
- Push Cancellation (Initiator to PPG)
- Status Query (Initiator to PPG)
- Client Capabilities Query (Initiator to PPG)

8.3 Push Submission

The Push message contains three entities: a control entity, a content entity, and optionally a capability entity. These are bundled together in a multipart/related message, which is sent from the Push Initiator to the PPG.

The control entity is an XML document that contains delivery instructions destined for the PPG, whereas the content entity is destined for the mobile device. The PPG may or may not convert this content into a more bandwidth-optimized form before forwarding it over-the-air: WML could be encoded to WML-C, while other content types will be completely unknown to the PPG or even encrypted for use by the mobile device only.

The optional capability entity contains the client capabilities that the message was formatted for, in UAPROF [UAPROF] format. The Push Initiator may create this entity to indicate what it *thinks* the client capabilities are.

8.4 Confirmation Notification

If the Push Initiator has requested confirmation of successful delivery, this message is transmitted from the PPG to the Push Initiator when the content has been delivered to the mobile device (over a 2-way bearer) or transmitted to the device (over a 1-way bearer). It contains an XML entity. It is also transmitted in case of a detected delivery failure, informing the Initiator thereof.

One key feature of the Push Framework is the possibility for a Push Initiator to rely on the response from the PPG: a confirmed push is confirmed by the WAP device when (and only when) the target application has taken responsibility for the pushed content. If it cannot take that responsibility, it must abort the operation, and the Push Initiator will know that the content never reached its destination. There are no loopholes within the framework for false positive delivery confirmations.

8.5 Push Cancellation

This is an XML entity transmitted from the Push Initiator to the PPG, requesting cancellation of previously submitted content. The PPG responds with an XML entity whether or not the cancellation was successful.

8.6 Status Query

This is an XML entity transmitted from the Push Initiator to the PPG, requesting status of previously submitted content. The PPG responds with an XML entity.

8.7 Client Capabilities Query

This is an XML entity transmitted from the Push Initiator to the PPG, requesting the capabilities of a particular device on the network (see also *Client Capabilities Query*, section 7.6). The PPG responds with a multipart/related in two parts, where the multipart root is the result of the request, and the second part is the capabilities of the device in the format defined by the User Agent Profiles group (see [UAPROF]).

8.8 HTTP Tunnelling

The PAP is carried over HTTP/1.1 in this issue of WAP Push. HTTP POST is used to transmit the information, and the HTTP transaction always returns result code 202 (“accepted for processing”) when the HTTP transaction itself succeeds; the response PAP packet may contain a PAP error, though. Compare how SMTP carries mail message headers ([RFC821], [RFC822]). See [HTTP] for more information about HTTP/1.1.

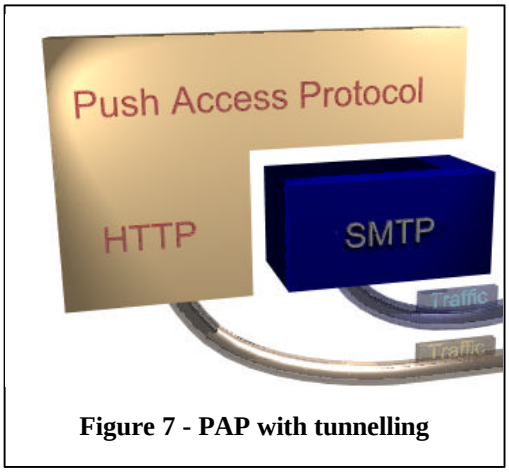

Figure 7 - PAP with tunnelling

Future versions may support other carrier protocols, like SMTP. The figure at the right illustrates the model: the framework has been built to easily incorporate other protocols, where the PAP could use these for tunnelling instead of or parallel to HTTP.

8.9 Reference

For more information, see [PushPAP].

9. The Service Indication

The *Service Indication* (SI) content type provides the ability to send notifications to end-users in an asynchronous manner. Such notifications may, for example, be about new e-mails, changes in stock price, news headlines, advertising, reminders of e.g. low prepaid balance, etc.

In its most basic form, an SI contains a short message and a URI indicating a service. The message is presented to the end-user upon reception, and the user is given the choice to either start the service indicated by the URI immediately, or postpone the SI for later handling. If the SI is postponed, the client stores it and the end-user is given the possibility to act upon it at a later point of time.

9.1 Reference

For more information, see [SI].

10. The Push Over-The-Air Protocol

The Push Over-The-Air (OTA) protocol is a thin protocol layer architecturally on top of WSP. It is this part of the Push Framework that is responsible for transporting content from the PPG to the client and its user agents.

The OTA protocol may use WSP sessions to deliver its content. At most one simultaneous push WSP session is established for each combination of OTA server , user agent, OTA client , and WDP bearer.

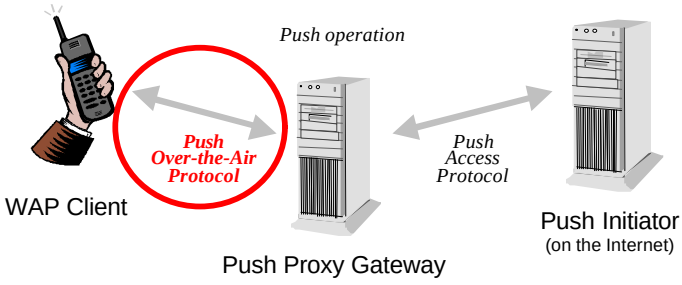

Figure 8 - OTA highlighted

A problem with making reliable pushes is that any confirmed push needs an active WSP session, and these cannot be initiated from the server: the client establishes WSP sessions, not the server. To solve the case where there is no active WSP session, the Push framework introduces a Session Initiation Application in the client, that listens to session requests from the OTAservers and responds by setting up a WSP session for push purposes. See section 11.1.

The client may verify the identity information in this request against a list of well-known OTAservers before attempting to establish the sessions to be used as conduits for the push traffic.

Push delivery may also be performed without the use of sessions, in a connectionless manner. This is needed in one-way networks.

10.1 Reference

For more information, see [PushOTA].

11. The Client-Side Infrastructure

The Push Framework places some additional demands on the client.

11.1 The Session Initiation Application (SIA)

A confirmed push requires an active WSP session. Only the client can initiate sessions. Therefore, if the server gets a request for a confirmed push to a client, and there is no active session to that client, there is theoretically nothing the server can do about it: the server needs a session to do the push, but only the client can set one up. The action from the Push Framework to circumvent this limitation is to send a session initiation request, *without* and *outside* an active WSP session (using connectionless push), to a special application in the client known as the Session Initiation Application (SIA). This request contains information about the real target client application.

As the SIA in the client receives a service initiation request, it establishes a session with the PPG, saying which applications accept content over the newly opened session. The SIA may also ignore the request if there is no suitable installed application for the content. See section 12.1.4 and [PushOTA] for some more information about this behaviour.

11.2 The Application Dispatcher

When a client receives pushed content, this is examined by a *dispatcher* that looks at the content and examines it to determine its destination application. This dispatcher is responsible for rejecting content that does not have a suitable destination application installed, and for confirming push operations to the PPG when the appropriate application takes responsibility for pushed content. See Application-Level Addressing, section 12.1.

12. Addressing scheme

When addressing a Push transaction, addressing may be done on several levels. Overall, a Push address takes the form of

WAPPUSH=Address /type=foo @ppg.mobile.net (without the whitespace)

The *type* switch (which is always present) indicates the type of address. For more information about the available values for *type*, see [PushPPG], "Addressing" section.

Addressing may be done on a *device* level, a *user* level and an *application* level. Of these three, the former two use the address-with-switch syntax, whereas the application-level addressing is a bit more complex.

The *ppg.mobile.net* part is the Internet host name of the Push Proxy Gateway.

12.1 Application-Level Addressing

A Push Initiator may target a specific user agent in the device for its content. To identify this user agent, its developer assigns it a URI that is under the developer's control; this URI is then transmitted along with the content to the client, where this content is dispatched to the correct user agent.

12.1.1 OTA Efficiency and Numeric Identifiers

To improve Over-the-Air efficiency, numeric identifiers may be used instead. WINA [WINA] will assign numbers to frequently-used user agents such as WAE and WTA, to avoid the overhead of a long URI being sent OTA.

Lists of assigned numeric identifiers are published by WINA. If a PPG is requested to push content with an application address URI that it recognises as a URI that has a numeric counterpart, the URI is replaced with this binary compact identifier OTA.

The push initiator may itself request a binary identifier to be used, an identifier that is not registered. This is discouraged with deployed applications because of the possibility of collisions; it is mainly intended for experimental user agents that have not been publicly deployed yet.

12.1.2 The PPG Logic

The logic in the PPG when determining which user agent identity to use goes like this: the Push Initiator specifies an application URI, an application numeric code, or both. If both are specified, the binary value takes precedence over the URI; if a registered binary value exists for a specified URI, that value takes precedence over both.

The Push Initiator may specify this user agent identity in the content headers.

12.1.3 WSP Connectionless Mode

In WSP connectionless mode, there are two *dispatcher ports* in the client, one secure and one unsecure, that accept pushes and forward them to the appropriate user agent based on the application id. All connectionless pushes go to one of these two client ports.

12.1.4 The Session-Based Case

The OTA server, in doing an OTA Push, connectionlessly pushes a Session Initiation Request (SIR) that contains the application id to a Session Initiation Application (SIA) in the client. The client responds by failing the session initiation or by setting up a session, indicating to the OTA server for which application ids the new session accepts pushed content. The client may also set up a "wildcard" session that accepts "…and everything else".

It should be pointed out that push sessions may coincide with pull sessions set up by the client; the client may indicate push capability when setting up a WSP session.

The Push Framework makes no implications as to which implementation is preferred, as long as confirmed and unconfirmed pushes are properly supported.

12.1.5 Example

Let's assume a PPG has content for client Foo, for an application called Spitzensparken. In addition, the Push Initiator has requested that this push be confirmed. There's currently no session between the PPG and Foo, so the PPG needs to establish one to use the confirmed-push WSP services.

The PPG constructs a session initiation request for Foo, indicating it wants to push reliably to application Spitzensparken. This request is sent to the Session Initiation Application at Foo, connectionlessly, just like any other content. The client's dispatcher gets the request, sees it's for the SIA, and sends it onward. The SIA, on receipt of this request, checks if the target application is installed in Foo and the user preferences. It notes that application Spitzensparken is, in fact, installed on this client, so the client should set up a session to accept the push.

Now, the owner of this particular device doesn't want to expose what applications he's got installed in the device to anybody over any network. The SIA notes this and sets up a WSP session with the PPG, saying that the session accepts content for any application; if the user had been less paranoid, applications for which this session could be used would have been explicitly listed.

Once the session has been set up, the PPG does the confirmed push over that session, and the client's dispatcher gets the content packet that started the operation. The dispatcher gets the packet, sees that it is for application Spitzensparken, and passes it to this application. When (and only when) the Spitzensparken application takes responsibility for the content packet (usually, after it has succeeded in copying the content packet into memory under the application's control), the push is confirmed all the way back to the Push Initiator.

12.2 Reference

For more information, see [PushPPG], "Addressing" section.

13. Security Considerations

When implementing WAP Push, security and trust models come into consideration in several areas. These are examples of questions that may arise;

- How can a PI be authenticated?
- What role could the PPG play in the security and trust model?
- What are the access control policies for a Push Initiator and pushed content?
- How can a client authenticate something if it has no certificate?

Regardless of these issues, it should be kept in mind that the Push Framework is always capable of providing the client with enough information to have a trust model and security policy of its own.

13.1 Authenticating a Push Initiator

It is important that a Push Initiator is authenticated in one form or another, depending on the security environments in which the Push Initiator and PPG are operating. This is an attempt to list some of the possible solutions.

- **Use of Session-level Certificates (TLS, SSL).**
  If the network between the Push Initiator and PPG is not trusted (e.g., the Internet, a very large Intranet, etc.), TLS/SSL can be used between the Push Initiator and the PPG.
- **Use of Object-level Certificates (Signed and/or Encrypted Content).**
  Certificates could be used to sign and/or encrypt the pushed content on an end-to-end basis. This would not involve the PPG in the authentication process, and would strengthen confidence in the content authenticity at the client end of the push submission.
- **HTTP Authentication.**
  Even though the most common form of HTTP authentication is the basic authentication (i.e., a user/password pair), other forms of HTTP authentication (e.g., digest based) might be preferable. The major differences between this approach and the use of certificates are that the latter is stronger in scalability and confidence, while the former is weak in these aspects.
- **A Combination of Technologies.**
  Technologies could be combined. For example, the Push Initiator can establish an anonymous TLS/SSL session with a PPG, whereupon HTTP authentication could be used to authenticate the Push Initiator. Signed and/or encrypted content could then be sent over this authenticated session.
- **Trusted Network.**
  In many real world installations, the network between the Push Initiator and the PPG is a private network. Therefore, the Push Initiator is implicitly trusted in such installations.

13.2 Client Delegation of Authentication

This refers to the principle that trust can be transitive. If a client and a PPG can establish trust, the PPG can authenticate a Push Initiator on behalf of that client. The trust can be established between a client and a PPG by keeping a list of trusted PPG's in a client. The list could also be assumed *a priori*.

After a PPG has been authenticated by a client, the client could look up its list of trusted PPG's. If the PPG is listed as trusted, the client can trust the PPG. Note that WAP has already solved the issue of authentication between a WAP client and a WAP server [WTLS].

Using the methods described in the previous section, a PPG can authenticate a Push Initiator with various levels of confidence. If it does, it could indicate in the push headers if the origin server is authenticated. The origin of the content, in turn, may be indicated by using the Content-Location header or any other location related element in the push content (e.g., URI in SI content).

There's also a header field for a PPG to indicate to the client if the entire content should be trusted, even if there is no Content-Location header or other location related element in the push content.

Needless to say, this delegation model does not exclude the use of a Push Initiator's own certificate or public key for end-to-end authentication of the origin server (all the way to the client).

13.3 Possible PPG Filtering and Access Control

The PPG can perform filtering and access control to discard pushed content that originates from an untrusted or unauthorised origin server. Such a feature is left to the discretion of the PPG implementer and the business relationship between the WAP service subscriber and the mobile network operator.

14. Scopes of the different Push documents

- **Architectural Overview**
  This document. The purpose of it is to be a starting point for anybody wanting to know more about the WAP Push technology, before taking on the other specifications.
- **Push Access Protocol specification**
  This document specifies the protocol with which a Push Initiator contacts the PPG. See section 8 for a brief description.
- **Push Proxy Gateway Service specification**
  This document specifies the Push Proxy Gateway functionality, and how it interacts with the Push Access Protocol and the Push Over-The-Air protocol. See section 7 for a brief description.
- **Push OTA Protocol specification**
  This document specifies the protocol with which a PPG talks to a Push-capable WAP client. See section 10 for a brief description.
- **Service Indication specification**
  This document specifies a content type used for notifying users they have new data waiting on aserver. See section 9 for a brief description.
- **Service Loading specification**
  This document specifies a content type that instructs the client to automatically load a URI.
- **Push Message specification**
  This document specifies end-to-end properties of a push message.