Push Over The Air

Approved Version 2.1 – 05 Apr 2011

Open Mobile Alliance

OMA-WAP-TS-PushOTA-V2_1-20110405-A

Continues the Technical Activities
Originated in the WAP Forum

Contents

Figures

1. Scope

This specification defines the Over the Air protocol for delivery of content to a mobile terminal from a Push Proxy Gateway (PPG), referred to as Push OTA protocol. The protocol specified in this document is an application layer protocol that can run on top of the WSP protocol [WSP], or on top of HTTP 1.1 [RFC2616].

The Push OTA protocol specified in this document addresses the following considerations:

- means for server initiated asynchronous push.
- means for application addressing.
- means for exchange of push control information over the air.
- means for bearer selection and control.
- means for authentication.

2. References

2.1 Normative References

[ClientID] "Client ID Specification". Open Mobile Alliance™.
WAP-196-ClientID. URL: http://www.openmobilealliance.org/

[IOPProc] “OMA Interoperability Policy and Process”. Open Mobile Alliance™. OMA-IOP-Process-v1_0.
URL:http//www.openmobilealliance.org/

[ProvCont] "Provisioning Content Type Specification". Open Mobile Alliance™.. WAP-183-ProvCont.
URL: http://www.openmobilealliance.org/

[PushMsg] "Push Message Specification". Open Mobile Alliance™.
WAP-251-PushMessage. URL: http://www.openmobilealliance.org/

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

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell.
November 1997. URL:http://www.ietf.org/rfc/rfc2234.txt

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

[RFC2617] "HTTP Authentication: Basic and Digest Access Authentication". J. Franks et al.
June 1999. URL:http://www.ietf.org/rfc/rfc2617.txt

[SHA] "Secure Hash Standard". NIST FIPS PUB 180-1. National Institute of Standards and Technology, U.S. Department of Commerce. Draft May 1994.

[WAPTLS] "WAP TLS Profile and Tunnelling". Open Mobile Alliance™.
WAP-219-TLS. URL: http://www.openmobilealliance.org/

[WDP] "Wireless Datagram Protocol". Open Mobile Alliance™.
WAP-259-WDP. URL: http://www.openmobilealliance.org/

[WSP] "Wireless Session Protocol". Open Mobile Alliance™.
WAP-230-WSP. URL: http://www.openmobilealliance.org/

[W-HTTP] "Wireless Profiled HTTP". Open Mobile Alliance™.
WAP-229-HTTP. URL: http://www.openmobilealliance.org/

[W-TCP] "Wireless Profiled TCP". Open Mobile Alliance™.
WAP-225-HTTP. URL: http://www.openmobilealliance.org/

[WTLS] "Wireless Transport Layer Security Protocol". Open Mobile Alliance™.
WAP-261-WTLS. URL: http://www.openmobilealliance.org/

2.2 Informative References

[IANA] "Internet Assigned Numbers Authority", URL: http://www.iana.org/

[ProvArch] "WAP Provisioning Architecture Overview". Open Mobile Alliance™.
WAP-182-ProvArch. URL: http://www.openmobilealliance.org/

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

3. Terminology and Conventions

3.1 Conventions

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

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

3.2 Definitions

Application	A value-added data service provided to a 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 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, 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 address 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 be 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.
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"
Multicast Message	a push message containing a single address which implicitly specifies more than one 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 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 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.

Registration	refers to a procedure where the PPG becomes aware of the terminal's current capabilities and preferences.
Registration Context	a state where the PPG is aware of at least the last capabilities and preferences conveyed from the terminal.
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).
Terminal	see "client".
Terminal-ID	an identifier that is used by a PPG to uniquely identify a terminal.
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.

3.3 Abbreviations

ABNF	Augmented Backus-Naur Form
CPI	Capability and Preference Information
CSD	Circuit Switched Data
DNS	Domain Name Server
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
MSISDN	Mobile Station International Subscriber Directory Number
OMA	Open Mobile Alliance
OTA	Over The Air
OTA-HTTP	(Push) OTA over HTTP
OTA-HTTP-TLS	OTA-HTTP over TLS
OTA-WSP	(Push) OTA over WSP
PDP	Packet Data Protocol
PI	Push Initiator
PO-TCP	PPG Originated TCP connection establishment method
PPG	Push Proxy Gateway
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RFC	Request For Comments
SHA-1	Secure Hash Algorithm 1
SIA	Session Initiation Application
SIR	Session Initiation Request
SMS	Short Message Service
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TO-TCP	Terminal Originated TCP connection establishment method
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security

4. Introduction

The push architecture allows a PPG to send data to a terminal in an asynchronous manner. The PPG and the terminal communicate using the Push OTA protocol, which utilises WSP [WSP] and/or HTTP [W-HTTP] services. The first protocol variant is referred to as "OTA-WSP", and the latter as "OTA-HTTP".

Connection-oriented push requires that some point-to-point connectivity (a push session if OTA-WSP is used, a TCP connection if OTA-HTTP is used) be established before the push content can be delivered. Connectivity for connection-oriented push can be shared among multiple terminal applications. A terminal application is identified by its Application-ID.

Connectionless push is always performed using WSP/WDP. The two registered WDP ports for connectionless push can be shared among multiple terminal applications.

A PPG is able to request a terminal to initiate connectivity by sending a special content type to the terminal using connectionless push.

The overall push architecture is outlined in Figure 1.

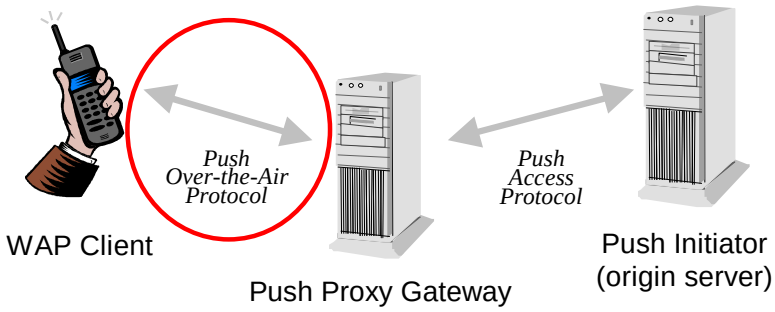

Figure 1: WAP Push Architecture

5. Protocol Variants

The Push OTA Protocol can be implemented to run on top of WSP [WSP], as described in section 6, or on top of HTTP 1.1 [RFC2616], as described in section 7. The WSP variant, henceforth referred to as "OTA-WSP", provides for both connection-oriented and connectionless push.

The HTTP variant, referred to as "OTA-HTTP", only provides functionality for connection-oriented push. If TLS [WAPTLS] is implemented in conjunction with OTA-HTTP to provide transport layer hop-by-hop security, this protocol variant is referred to as "OTA-HTTP-TLS".

A terminal MUST support either the connection-oriented services provided by OTA-WSP or those provided by OTA-HTTP when connection-oriented push is implemented, and MAY support both. A Push Proxy Gateway SHOULD implement both variants in order to support a wide range of mobile terminals. Both the terminal and the PPG MUST support the connectionless services provided by OTA-WSP as defined in section 6.2.1.

Note: The protocol variants use different ports, which are registered with IANA.

6. Push OTA Protocol over WSP (OTA-WSP)

This section describes how OTA-WSP is implemented. This variant runs on top of WSP [WSP], and is suitable for use with low-bandwidth bearers that do not support TCP/IP, e.g. SMS.

6.1 Service Primitive Definition

6.1.1 Notations

Notations for primitives and parameters follow the notations defined in [WSP].

6.1.2 Service Primitive Overview

This section is informative.

The primitives defined in this section include both push operational primitives and push management primitives. While the push operational primitives are used to deliver content from a server (also referred to as "PPG") to a client (also referred to as "terminal"), the push management primitives are used to establish and manage the push session.

Figure 2 demonstrates the layer-to-layer communication through the primitives.

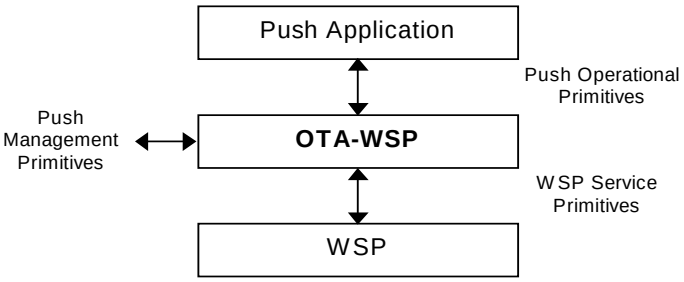

Figure 2: Illustration of Layer to Layer Communication

6.1.3 Push Operational Primitives

6.1.3.1 Po-Push

This primitive is used to send information from the server in an unconfirmed manner on a push session using the connection-oriented service.

Primitive / Parameter	Po-Push	
	req	***ind***
Push Headers	O	C(=)
Authenticated	O	C(=)
Trusted	O	C(=)
Last	O	C(=)
Push Body	O	C(=)

Push Headers are defined in [PushMsg].

Authenticated indicates if the initiator URI is authenticated by the server.

Trusted indicates if the push content is trusted by the server. This provides a mechanism for a client to delegate its trust policy to the server (i.e. PPG).

Last is a hint to the client that this is the last message to send according to the server's best knowledge. The client MAY terminate use of the network bearer.

Push Body is the content in the push, which is semantically equivalent to an HTTP entity body. If *Push Body* is empty, the rest of the parameters MUST be inspected and used (e.g. for bearer or cache control), if applicable before the empty *Push Body* is ignored.

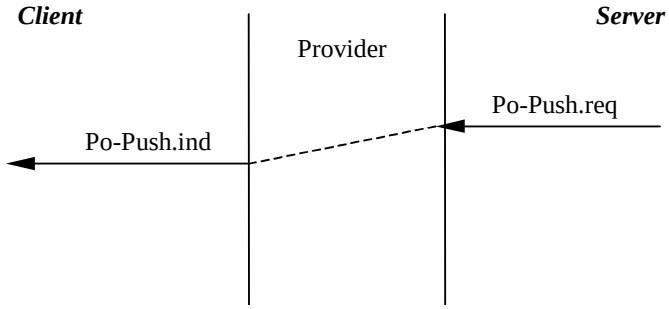

Figure 3: Unconfirmed Push

6.1.3.2 Po-ConfirmedPush

This primitive is used to send information from the server in a confirmed manner on a push session using the connection-oriented service. It is the service user (e.g. client push application) that confirms the push by invoking Po-ConfirmedPush.res primitive when the service user takes responsibility for the push message. If the service user can not take responsibility for the push message, it MUST abort the push operation by invoking the Po-PushAbort.req primitive (6.1.3.3). The service provider MAY abort the push on behalf of the service user at its discretion (e.g. if the service user does not respond).

Primitive / Parameter	Po-ConfirmedPush			
	req	***ind***	***res***	***cnf***
Server Push Id	M	–	–	M(=)
Client Push Id	–	M	M(=)	–
Push Headers	O	C(=)	–	–
Authenticated	O	C(=)	–	–
Trusted	O	(=)	–	–
Last	O	C(=)	–	–
Push Body	O	C(=)	–	–
Acknowledgement Headers	–	–	O	P(=)

Server Push Id is defined in S-ConfirmedPush primitive in [WSP].

Client Push Id is defined in S-ConfirmedPush primitive in [WSP].

Push Headers are defined in [PushMsg].

Authenticated indicates if the initiator URI is authenticated by the server.

Trusted indicates if the push content is trusted by the server. This provides a mechanism for a client to delegate its trust policy to the server (i.e. PPG).

Last is a hint to the client that this is the last message to send according to the server's best knowledge. The client MAY terminate use of the network bearer.

Push Body is the content in the push, which is semantically equivalent to an HTTP entity body. If *Push Body* is empty, the rest of the parameters MUST be inspected and used (e.g. for bearer or cache control), if applicable before the empty *Push Body* is ignored.

Acknowledgement Headers is defined in S-ConfirmedPush primitive in [WSP].

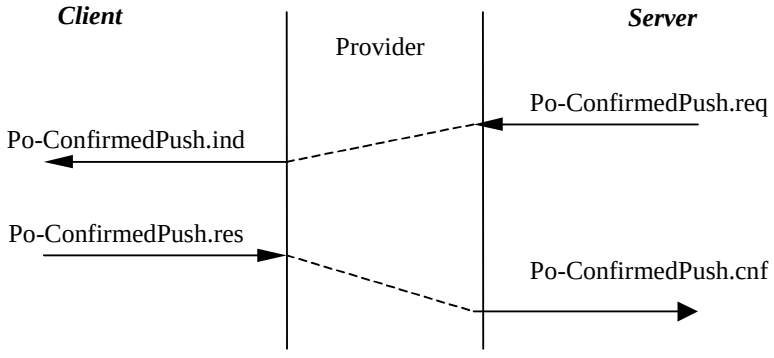

Figure 4: Confirmed Data Push

6.1.3.3 Po-PushAbort

This primitive is used to reject a push operation. It is part of the *ConfirmedPush* facility. It is mapped directly to S-PushAbort primitive in [WSP]. Only the following values for the *Reason* parameter SHOULD be used:

Name	Description
USERREQ	aborted without specific causes, retries allowed
USERRFS	aborted without specific causes, no retries
USERPND	aborted because the push message can not be delivered to the intended destination
USERDCR	aborted because the push message is discarded due to resource shortage
USERDCU	aborted because the content type can not be processed

6.1.3.4 Po-Unit-Push

This primitive is used to send information from the server to the client in a unconfirmed manner on the connectionless session service [WSP].

Primitive / Parameter	Po-Unit-Push	
	req	***ind***
Client Address	M	M(=)
Server Address	M	M(=)
Push Id	M	M(=)
Push Headers	O	C(=)
Authenticated	O	C(=)
Trusted	O	C(=)
Last	O	C(=)
Push Body	O	C(=)

Client Address identifies the peer to which the push is to be sent.

Server Address identifies the originator of the push.

Push Id MAY be used by the service users to distinguish between pushes.

Push Headers are defined in [PushMsg].

Authenticated indicates if the initiator URI is authenticated by the server.

Trusted indicates if the push content is trusted by the server. This provides a mechanism for a client to delegate its trust policy to the server (i.e. PPG).

Last is a hint to the client that this is the last message to send according to the server's best knowledge. The client MAY terminate use of the network bearer.

Push Body is the content in the push, which is semantically equivalent to an HTTP entity body. If *Push Body* is empty, the rest of the parameters MUST be inspected and used (e.g. for bearer or cache control), if applicable before the empty *Push Body* is ignored.

6.1.4 Push Management Primitives

6.1.5 Pom-Connect

This primitive is used to create a push session as requested by the client. It is mapped to S-Connect primitive in WSP [WSP] with additional parameters.

Primitive / Parameter	Pom-SessionCreate			
	req	***ind***	***res***	***cnf***
Server Address	M	M(=)	–	–
Client Address	M	M(=)	–	–
Client Headers	O	C(=)	–	–
Requested Capabilities	O	M	–	–
Server Headers	–	–	O	C(=)
Negotiated Capabilities	–	–	O	C(=)
Accept Application	O	C(=)	–	–
Bearer Indication	O	C(=)	–	–

Server Address identifies the server with which the push session is to be established.

Client Address identifies the client that will receive the push content.

Client Headers, *Server Headers*, *Requested Capabilities*, and *Negotiated Capabilities* are defined in S-*Connect* primitive [WSP].

Accept Application provides a list of *ApplicationID*s. The first Application in the list identifies the default Application-ID. If the list is empty, or if the first element in the list cannot uniquely identify an application (e.g. *), WML User Agent is assumed the default Application-ID.

Bearer Indication indicates the bearer type over which the push session is established. The service user (e.g. server) MAY use the information to make bearer selection decisions. Use the well-known bearer type codes as defined in an appendix of [WDP].

6.1.5.1 Pom-Suspend

This primitive is used to request the push session to be suspended so that no activity is allowed. This primitive is mapped directly to S-Suspend primitive in WSP [WSP].

6.1.5.2 Pom-Resume

This primitive is used to request the push session, which is previously suspended, to be resumed. It is mapped directly on S-Resume primitive in WSP [WSP].

Primitive / Parameter	Pom-Resume			
	req	***ind***	***res***	***cnf***
Server Address	M	M(=)	–	–
Client Address	M	M(=)	–	–
Client Headers	O	C(=)	–	–
Server Headers	–	–	O	C(=)
Bearer Indication	O	C(=)	–	–

Server Address identifies the server with which the push session is to be established.

Client Address identifies the client that will receive the push content.

Client Headers and *Server Headers* are defined in [WSP].

Bearer Indication indicates the bearer type over which the push session is established. The service user (e.g. server) MAY use the information to make bearer selection decisions. Use the well-known bearer type codes as defined in an appendix of [WDP].

6.1.5.3 Pom-Disconnect

This primitive is used to terminate a push session. It is mapped directly on S-Disconnect primitive in WSP [WSP].

6.1.5.4 Pom-SessionRequest

This primitive is used by the server to request a push session to a client.

Primitive / Parameter	Pom-SessionRequest	
	req	***ind***
Client Address	M	M(=)
Server Address	M	M(=)
Push Headers	M	M(=)
SIA Content	M	M(=)

Client Address identifies the peer to which the session is requested.

Server Address identifies the address of the server.

Push Headers are defined in S-Push primitive in [WSP]. It contains at least the following two headers,

- `Content-Type: application/vnd.wap.sia`
- `X-Wap-Application-Id: x-wap-application:push.sia`

SIA Content contains a list of Application-ID's required for push sessions and a list of contact points. It is a special content type as defined in section 8.

6.2 Protocol Description

OTA-WSP provides for *connectionless* and *connection-oriented* push as described in subsequent sections.

6.2.1 Connectionless Push

The connectionless push must be performed through WSP S-Unit-Push [WSP], which is one of WSP connectionless session service primitives. Two registered WDP ports [WDP], secure and non-secure ports, are reserved in every client capable of connectionless push. The client MUST support the non-secure port and MAY support the secure port. If the secure port is supported, WTLS MUST be supported on the port [WTLS]. To accommodate server initiated WTLS connections, a client supporting secure connectionless push MUST be able to initiate the WTLS negotiation process as a result of receiving a Hello Request message [WTLS] on its registered secure WDP port. In doing so, the client MUST use the address quadruplet from where the Hello Request was originated. To protect against spoofing, the client SHOULD validate the Hello Request by comparing the source address from where the Hello request was originated with a pre-existing list of contact points for trusted servers. The client SHOULD ignore the Hello Request if the validation fails.

6.2.2 Connection-Oriented Push

The connection-oriented push MUST be performed through the WSP S-Push (e.g. unconfirmed push) or S-ConfirmedPush of WSP session service primitives. A push session MUST be established to carry out those primitives. A push session is a WSP session on which the confirmed, unconfirmed or both, push capability(ies) [WSP] is enabled.

The push session can use either secure or non-secure transport services. Server-side port numbers are reserved in [WDP] for both options. WTLS MUST be used if the secure transport service is required. The secure transport service is required if either the port number in a contact point is a registered secure port [WDP] or the secure transport is indicated in a pre - existing list of contact points for PPGs.

6.2.3 Application Addressing

The push content can be delivered to any application, as identified by the Application-ID, in a client. In the case of connectionless push, the push content is first delivered to one of the registered WDP ports in the client, the Push OTA-WSP layer of the client is responsible to further deliver the push content to the application as identified by the Application-ID. For the connection-oriented push, the push content is first delivered on the push session, the Push OTA-WSP layer of the client is responsible to further deliver the push content to the application as identified by the Application-ID.

The default Application-ID is that of the WML User Agent for connectionless push, and also for connection-oriented push unless another value is negotiated during push session establishment.

6.2.4 Initiator Authentication

Push initiator authentication by the PPG may be indicated to the terminal through the inclusion of the *Authenticated* flag and *Initiator URI*. That model of authentication is based on transitive trust established between the PI and the client. The PPG MUST positively authenticate the PI to use the *Authenticated* flag.

When receiving the *Authenticated* flag, the client MAY determine if the push initiator is trusted by comparing its list of trusted initiator URIs with the authenticated *Initiator URI* in the push message.

6.2.5 Trust Delegation

The PPG MAY include the *Trusted Flag* to indicate to the client that the content is trusted based on its best knowledge. The client MAY trust the content if it has a pre-existing trust relationship with the Push Proxy Gateway.

6.2.6 Bearer Selection and Control

Bearer Type Indication provides means for the client to report the actual bearer used on a particular session when the bearer type cannot be inferred otherwise. The PPG MAY use this information to support bearer selection in an implementation dependent manner.

Bearer control is facilitated with the *Last* flag, whose presence provides a hint to the client that this is the last message to send according to the server's best knowledge. The client MAY terminate use of the network bearer.

6.3 Protocol Operations

6.3.1 Application Dispatching

When a client receives a push, it uses the Application-ID to locate an application as identified by the Application-ID. For connectionless push, the client dispatches the push content received on the registered WDP port to the application identified by the Application-ID. For connection-oriented push, the client dispatches the push content received on the push session to the application identified by the Application-ID.

6.4 Protocol Data Unit Definition

This section describes the protocol data units to be used with OTA-WSP.

6.4.1 Header Based Protocol Data Unit

The header definition rules in this sub-section follow the rules in the HTTP [RFC2616] and ABNF [RFC2234] specifications. WSP compact encoding rules MUST be used to encode them for over the air efficiency.

6.4.1.1 Accept-Application

```
Accept-Application = "Accept-Application" ":" app-ranges
App-ranges = ( #app-id | "*" )
; app-id as defined in [PushMsg]
; * means any Application-ID.
```

6.4.1.2 Bearer-Indication

```
Bearer-Indication = "Bearer-Indication" ":" bearer-type
Bearer-type = 2HEXDIG
; Bearer-type as defined in [WDP].
```

6.4.1.3 Push Flag

```
Push-Flag = "Push-Flag" ":" 1*7BIT
; bit mask flags to indicate the following:
;     1: initiator URI is authenticated.
;    10: content is trusted.
;   100: last push message.
; other: reserved for future use.
```

7. Push OTA Protocol over HTTP (OTA-HTTP)

7.1 Protocol Overview

This section is informative.

This section describes how OTA-HTTP is implemented. It is designed to run on top of HTTP 1.1 [RFC2616], and is intended to be used in conjunction with bearers that support TCP/IP, e.g. GPRS. Due to the characteristics of TCP/IP and HTTP, only connection-oriented push is supported. Connectionless push is accomplished using WSP (see section 6.2.1).

Since this protocol variant relies upon the existence of an HTTP server in the mobile device, the device is not referenced as "client" in the subsequent sections, but instead "mobile terminal" or simply "terminal" to avoid confusion.

The core features of OTA-HTTP consists of:

- *IP connectivity procedure*
  The protocol is designed to work with mobile networks that support network initiated IP connectivity establishment procedures, and also with networks that solely rely upon the terminals' ability to establish IP connectivity with the network.

- *TCP connection procedure*
  In order to accommodate various mobile network types, e.g. with respect to IP address awareness (for example, a dynamically assigned terminal IP address might not always be known by the PPG), the protocol provides two methods for establishing the TCP connection to be used for communication between the PPG and the terminal.

- *Registration*
  The term "registration" refers to a procedure where the PPG becomes aware of the terminal's current capabilities and preferences. The information is conveyed using headers, and may be stored in the PPG to avoid that the information is communicated in future transactions. The registration procedure is always initiated by the PPG.

- *Content push*
  Delivery of content from the PPG to the terminal is accomplished by using HTTP's POST method. Hence, OTA-HTTP relies upon the existence of an HTTP server in the terminal, and an HTTP client in the PPG.

In addition to the functions listed above, OTA-HTTP provides a means to identify, and optionally authenticate, both the PPG and the mobile terminal during registration and push delivery. TLS may be used to provide additional authentication, data integrity, and confidentiality. The term "OTA-HTTP-TLS" is used when OTA-HTTP is used in conjunction with TLS to provide measures for secure push.

A mechanism for version control is also specified to allow future extensions of the protocol.

7.2 Protocol Description

7.2.1 HTTP Compliance

A terminal implementing OTA-HTTP MUST implement the HTTP server features specified for a "WAP Terminal" in [W-HTTP]. A PPG implementing OTA-HTTP MUST implement the HTTP client features specified for a "WAP Proxy" in [W-HTTP].

7.2.2 TLS Compliance

A terminal implementing OTA-HTTP-TLS MUST implement the TLS client features specified by [WAPTLS]. A PPG implementing OTA-HTTP-TLS MUST implement the TLS server features specified by [WAPTLS]. However, note that OTA-HTTP-TLS is OPTIONAL both for terminals and PPGs.

7.2.3 IP Connectivity Procedure

This section is informative.

Before a TCP connection between the PPG and the terminal can be established, the terminal needs to have IP connectivity with the network. For example, a circuit must be established for CSD, or a PDP context must be created for GPRS. The terminal's IP address does also need to be made known to the PPG.

In some networks it is possible to initiate the IP connectivity establishment procedure from the network, and by some means find out what IP address the terminal has been assigned (e.g. if static IP addresses are used, or by lookup in a RADIUS server).

In contrast, some networks do not offer the kind of functionality described in the previous paragraph, or it might not be available to the PPG. In that case, the PPG can send an SIR (see section 8) to the terminal using either connectionless push over a bearer where a well-known address can be used (e.g. MSISDN for SMS), or by using connection-oriented push if applicable.

It is also possible that the terminal takes the initiative of its own accord to establish IP connectivity with the network and then establish a TCP connection towards the PPG. In that case the PPG does not need to send an SIR to the terminal.

7.2.4 TCP Connection Procedure

In order to provide flexibility, OTA-HTTP offers two methods for establishing one or more TCP connections to be used for registration and push delivery (such TCP connections are henceforth referred to as "active TCP connections"). These are:

- ***T****erminal Originated* ***TCP*** *connection establishment method (****TO-TCP****)*
  This method provides the terminal with a means to establish a TCP connection towards the PPG that can be used for subsequent registration and push delivery.

- ***P****PG Originated* ***TCP*** *connection establishment method (****PO-TCP****)*
  This method provides the PPG with a means to establish a TCP connection towards the terminal that can be used for subsequent registration and push delivery.

The TCP connection methods listed above are further described in the subsequent section. Either the PPG or the terminal may at any time close an active TCP connection.

7.2.4.1 TCP Connection Methods

This section describes the methods available to establish an active TCP connection between the PPG and the terminal. The source port SHOULD be assigned from the range of dynamic ports [IANA]. The destination port MUST be the port specified for the method utilised.

A terminal implementing OTA-HTTP MUST support the non-secure, and MAY support the secure TO-TCP (OTA-HTTP-TLS) methods. It MUST also support the non-secure, and MAY support the secure PO-TCP (OTA-HTTP-TLS) methods.

A PPG implementing OTA-HTTP MUST support the non-secure, and MAY support the secure TO-TCP (OTA-HTTP-TLS) methods. It SHOULD also support the non-secure, and MAY support the secure PO-TCP (OTA-HTTP-TLS) methods.

7.2.4.1.1 The TO-TCP Method

This method allows a TCP connection established by the terminal towards the PPG to be used as the active TCP connection (this implies that the terminal must be prepared to receive HTTP requests on this connection). The destination port (in order of precedence) is:

- a port specified in SIR (if present)
- a provisioned port (if so provisioned) or another port agreed by some implementation specific means
- one of the registered push ports (non-secure/secure)

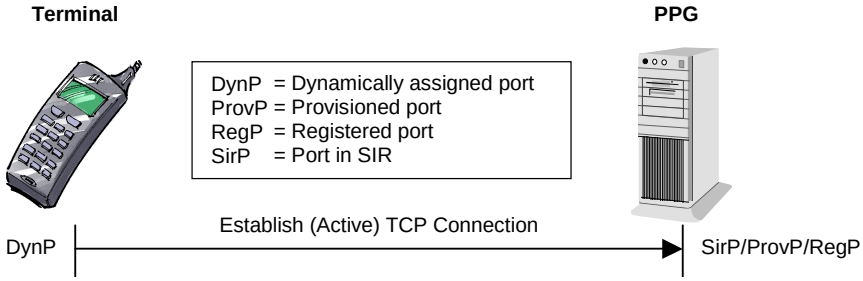

Figure 5: The TO-TCP method

If a terminal establishes a TCP connection towards the registered secure port on a PPG, or another port that requires TLS, the terminal MUST establish a TLS session on that connection before it accepts any push content via that connection.

7.2.4.1.2 The PO-TCP Method

This method assumes that the terminal has IP connectivity with the network (or that the PPG can initiate the IP connectivity establishment procedure via the network), and its IP address is known by the PPG. A TCP connection established by the PPG towards the terminal is used as the active TCP connection. The destination port (in order of precedence) is:

- a provisioned port (if so provisioned), or another port agreed by some implementation specific means
- one of the registered push ports (non-secure/secure)

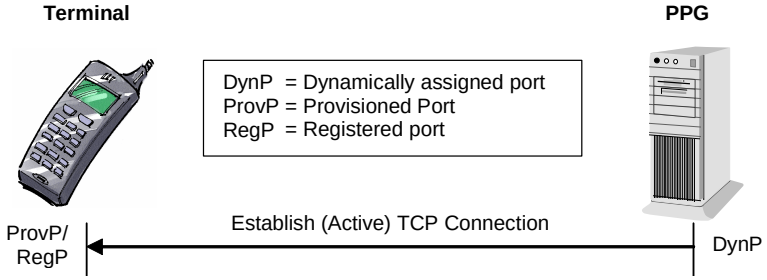

Figure 6: The PO-TCP method

If a PPG establishes a TCP connection towards the registered secure port on a terminal, or another port that requires TLS, a terminal supporting OTA-HTTP-TLS MUST establish a TLS session on that connection before it accepts any push content via that connection.

7.2.5 Terminal Registration

When an active TCP connection has been established between the PPG and the terminal, the PPG may at any time query the terminal for its capabilities and preferences. The push specific capability and preference information (CPI) are carried in a set of headers specified for this purpose. During this registration procedure, the terminal and the PPG are also identified and optionally authenticated (see section 7.2.6).

Once the CPI has been conveyed to the PPG, a *registration context* is established between the PPG and the terminal. The registration context is defined within the scope of a certain Terminal-ID, and also the bearer used when the CPI was conveyed. The CPI may change within the boundaries of a registration context. Each CPI is identified by a so-called CPITag, which is computed by the terminal, providing the PPG with a means to store multiple identifiable CPIs for a registration context.

The CPITag assumed to be valid by the PPG might be included in **registration requests** (using the HTTP OPTIONS method) made towards the terminal. If the assumed CPITag does not match the terminal's CPITag, or if it is not present, the terminal's current CPI and CPITag will be conveyed to the PPG by using headers specified for this purpose. On the other hand, if the CPITag matches, the information does not need to be conveyed.

Similarly, the CPITag assumed to be valid by the PPG might also be included in subsequent **push requests** (using the HTTP POST method) made towards the terminal. This provides a mechanism for **registration validation**. The terminal should reject the push request if the assumed CPITag does not match the terminal's CPITag. In this case the PPG will be made aware of the terminal's actual CPITag and can then, before it sends a new push request, either find the terminal's current CPI in its local storage or make a registration request if it is not found. On the other hand, if the CPITag matches, the push request is accepted and no further communication is needed in order to deliver the message.

The PPG SHOULD carry out the registration procedure when an active TCP connection has been established in order to identify/authenticate the terminal and find out about its capabilities and preferences.

7.2.5.1 Registration

PPG initiated registration is accomplished by sending an HTTP OPTIONS [RFC2616] request from the PPG to the terminal, using `/wappush` as Request-URI and an empty `HOST` header field. The `X-Wap-Push-ProvURL` header MAY be included in the request. See section 7.2.5.4 for further details.

The response from the terminal MUST, unless it rejects the request (e.g. if authentication is required), include the following headers if no `X-Wap-CPITag` header is conveyed from the PPG to the terminal:

- CPI headers (optional headers specified in section 7.2.5.4)
- the `X-Wap-CPITag` header

These headers MUST also be included in the response if a CPITag is conveyed from the PPG to the terminal and it does not match the terminal's current CPITag.

The assumed CPITag can be conveyed from the PPG to the terminal using either of these methods:

- include the CPITag in an SIR
- include the CPITag in the `X-Wap-CPITag` header in the OPTIONS request

The response to the OPTIONS request contains an HTTP [RFC2616] status code that reflects the outcome of that request (accepted, authentication required etc.). The `X-Wap-Push-Status` header (see section 7.4.2.1), indicating the outcome of the registration request, MUST be included in the response to the OPTIONS request.

The figure below shows an example of the procedure described above.

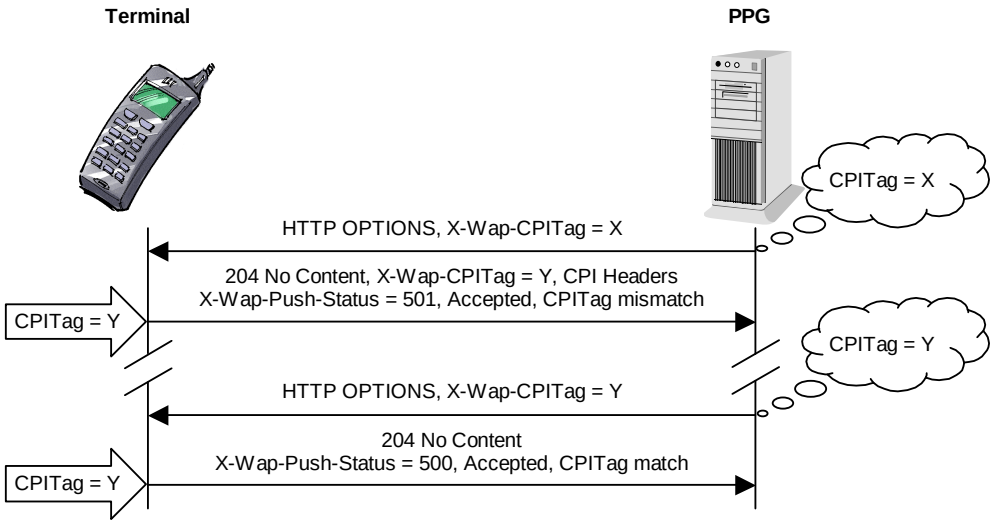

Figure 7: Registration (OPTIONS) request example

7.2.5.2 Registration Validation

This method is used in conjunction with delivery of push content between the PPG and the terminal using the HTTP POST method (see section 7.4), and is primarily used when the PPG assumes that the registration context it maintains contains the terminal's current CPI. The method can only be used when the PPG knows the coupling between the identity and the IP address of the terminal it is sending the POST request to, for example, if the PPG has performed the registration procedure, if static IP addresses are used, or if the PPG is able to communicate with some network entity that provides the coupling.

The terminal finds out if the PPG is aware of its current CPI by comparing its own CPITag with the assumed CPITag conveyed from the PPG to the terminal. The assumed CPITag can be conveyed using either of these methods:

- include the CPITag in an SIR
- include the CPITag in the `X-Wap-CPITag` header in the POST request

If the CPITag assumed by the PPG matches the terminal's current CPITag, the terminal MUST attempt to deliver the push content to the addressed application and respond as described in section 7.4.2, and the `X-Wap-CPITag` header MUST NOT be included in the POST response.

In contrast, if the assumed CPITag does not match the terminal's current CPITag, the terminal SHOULD silently discard the message body of the request (i.e. the push content). If the message body is discarded, the terminal MUST convey its CPITag to the PPG by including the `X-Wap-CPITag` header in the response. If it accepts the message body it SHOULD include the `X-Wap-CPITag` header in the response (see also the introduction to section 7.2.5 for additional explanation on CPI information lookup using the CPITag during registration validation).

If the assumed CPITag is not conveyed to the terminal, the terminal SHOULD accept the message body. If the message body is discarded the terminal MUST convey its CPITag to the PPG by including the `X-Wap-CPITag` header in the response. If it accepts the message body it SHOULD NOT include the `X-Wap-CPITag` header in the response.

The response to the POST request contains an HTTP [RFC2616] status code that reflects the outcome of that request (accepted, authentication required etc.). The `X-Wap-Push-Status` header (see section 8.4.2.1), indicating the outcome of the push/validation request, MUST be included in the response to the POST request.

The figure below shows an example of the procedure described above.

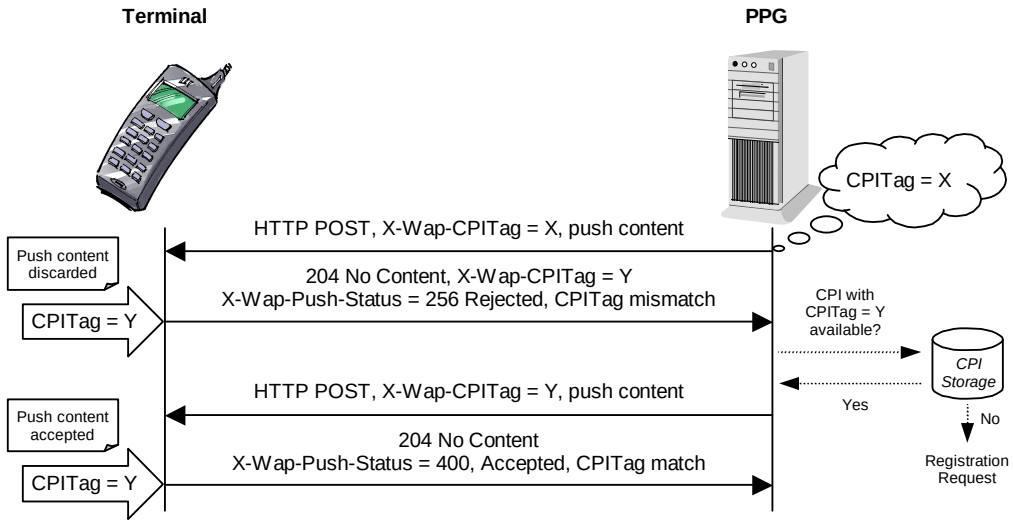

Figure 8: Registration validation (POST) example

7.2.5.3 The X-Wap-CPITag Header

As mentioned above, a specific CPITag value is used to represent a specific set of CPI header values. Each time one or more CPI headers change the terminal MUST re-compute the CPITag before it is conveyed to the PPG. The CPITag is carried in the `X-Wap-CPITag` header discussed in section 7.2.5.1 and 7.2.5.2. The ABNF [RFC2234] format of the header is:

```
X-Wap-CPITag = "X-Wap-CPITag" ":" CPItag
CPItag = 4*OCTET
```

The `CPItag` value is a four octet truncated hash of the CPI, and MUST be computed as follows:

- concatenate all CPI header (see section 7.2.5.5) values that are sent in the response
- apply a hashing algorithm that generates at least a four octet hash on the concatenated value. The SHA-1 [SHA] algorithm is RECOMMENDED.
- use the first four octets of the output
- generate the `CPITag` by base64-encoding these four octets

This specification does not specify how the CPI header concatenation should be done. However, a terminal SHOULD ensure that it is done in a consistent manner so that a certain set of CPI header values always results in the same concatenated value (and thereby the same CPITag). For example, if $CPITag_1$ represents the terminal's CPI when English is selected as the most preferred language, and the user switches to Swedish and thereby computes a new CPITag, a succeeding CPITag computation should result in $CPITag_1$ if the user chooses to switch back to English (assuming all other CPI headers remaining unaltered).

7.2.5.4 The X-Wap-Push-ProvURL Header

If the PO-TCP method was used to establish the active TCP connection, and the terminal supports WAP Provisioning [ProvArch], this OPTIONAL request header provides the PPG with a means to inform the terminal about which configuration context it should use (to obtain the appropriate Terminal-ID, authorization credentials, etc.) by indicating the configuration context's `ProvURL` [ProvCont].

A ProvURL value can be empty, which is indicated by including an empty X-Wap-Push-ProvURL header in the request. A terminal MUST NOT interpret the absence of the X-Wap-Push-ProvURL header as if a configuration context with an empty ProvURL value is indicated. If the header is absent it is left to the discretion of the terminal to select a suitable configuration context, or use other means to obtain appropriate configuration parameters.

The X-Wap-Push-ProvURL header does only need to be included in the first HTTP request sent towards the terminal within the scope of a specific active TCP connection established using the PO-TCP method. This allows the terminal to associate that active TCP connection with a certain ProvURL until the connection is closed.

The ABNF [RFC2234] format of the header is:

```
X-Wap-Push-ProvURL = "X-Wap-Push-ProvURL" ":" ProvURL
; ProvURL as defined in [ProvCont]
```

If the X-Wap-Push-ProvURL header is present, and the terminal supports WAP Provisioning, the following rules apply:

- If the specified ProvURL is non-empty and it matches one of the terminal's configuration contexts, the matching configuration context MUST be used. If no match can be found, the terminal MUST reject the request and return the appropriate status code (257 or 302) in the X-Wap-Push-Status header (see section 7.4.2.1).
- If the header is empty, it is left to the discretion of the terminal to select the appropriate configuration context among those having an empty ProvURL. If the terminal cannot find a provisioning context with an empty ProvURL, the terminal MUST reject the request and return the appropriate status code (257 or 302) in the X-Wap-Push-Status header (see section 7.4.2.1).

If the X-Wap-Push-ProvURL header is present, and the terminal does not support WAP Provisioning, the terminal MAY reject the request and return the appropriate status code (257 or 302) in the X-Wap-Push-Status header (see section 7.4.2.1).

7.2.5.5 CPI headers

The following sub-sections define the headers that are used to convey the terminal's CPI to the PPG as described in previous sections. All headers are OPTIONAL, and a terminal MAY include other headers among its CPI headers if it so wishes. If any of the CPI headers listed in the following sub-sections are not present in the response to a registration request, the PPG MUST assume their *default* values.

All header format definitions are expressed using ABNF [RFC2234].

7.2.5.5.1 X-Wap-Push-Accept

Header Name: X-Wap-Push-Accept

Description: List of supported content types that can be carried inside the application/http entity body (see section 7.4.1)

Format:
```
X-Wap-Push-Accept = "X-Wap-Push-Accept" ":" Accept-value
; Accept-value identical with HTTP's Accept header value
; [RFC2616]
```

Default: application/vnd.wap.sia, text/vnd.wap.si

7.2.5.5.2 X-Wap-Push-Accept-Charset

Header Name: `X-Wap-Push-Accept-Charset`

Description: List of supported content types character sets

Format:
```
X-Wap-Push-Accept-Charset = "X-Wap-Push-Accept-Charset"
                            ":" Charset-value
; Charset-value identical with HTTP's Accept-Charset header
; value [RFC2616]
```

Default: `UTF-8`

7.2.5.5.3 X-Wap-Push-Accept-Encoding

Header Name: `X-Wap-Push-Accept-Encoding`

Description: List of supported transfer encoding methods

Format:
```
X-Wap-Push-Accept-Encoding = "X-Wap-Push-Accept-Encoding"
                             ":" Encoding-value
; Encoding-value identical with HTTP's Accept-Encoding header
; value [RFC2616]
```

Default: `identity`

7.2.5.5.4 X-Wap-Push-Accept-Language

Header Name: `X-Wap-Push-Accept-Language`

Description: List of supported languages

Format:
```
X-Wap-Push-Accept-Language = "X-Wap-Push-Accept-Language"
                             ":" Language-value
; Language-value identical with HTTP's Accept-Language header
; value [RFC2616]
```

Default: `*`

7.2.5.5.5 X-Wap-Push-Accept-AppID

Header Name: `X-Wap-Push-Accept-AppID`

Description: List of applications the terminal supports, where each item in the list is an application-id in `absoluteURI` format as specified in [PushMsg]. A wildcard ("`*`") may be used to indicate support for any application (e.g. due to privacy concerns).

Format:
```
X-Wap-Push-Accept-AppID = "X-Wap-Push-Accept-AppID"
                          ":" (AppIDlist | "*")
AppIDlist = absoluteURI *("," SP absoluteURI)
```

Default: `*`

7.2.5.5.6 X-Wap-Push-MsgSize

Header Name: `X-Wap-Push-MaxMsgSize`

Description: Maximum size of a push message that the terminal can handle. Value is number of bytes.

Format:
```
X-Wap-Push-MaxMsgSize = "X-Wap-Push-MaxMsgSize" ":" *DIGIT
```

Default: `1400`

7.2.5.5.7 X-Wap-Push-Accept-MaxPushReq

Header Name: `X-Wap-Push-Accept-MaxPushReq`

Description: Maximum number of outstanding push requests that the terminal can handle

Format:
```
X-Wap-Push-Accept-MaxPushReq = "X-Wap-Push-Accept-MaxPushReq"
                               ":" *DIGIT
```

Default: `1`

7.2.5.5.8 X-Wap-Push-User-Agent Header

The X-Wap-Push-User-Agent response header field contains information about the server responding to the OPTIONS request originated by the PPG. This is for statistical purposes, the tracing of protocol violations, and the automated recognition of user agents for the sake of tailoring POST requests in order to avoid particular Terminal limitations in conjunction with CPI headers.

This header SHOULD be included in all OPTIONS responses to the PPG. Furthermore, if included, the field value MUST NOT be empty. The field can contain multiple product tokens and comments identifying the agent and any sub products that form a significant part of the user agent.

By convention, the product tokens are listed in order of their significance for identifying the application, [RFC2616].

```
X-Wap-Push-User-Agent = "X-Wap-Push-User-Agent" ":" 1*(product | comment)
```

Example:

```
X-Wap-Push-User-Agent: make/model
```

Note: the definition of user agent in this case is not the same as application identifier.

7.2.5.6 CPI and User Agent Profile

The X-Wap-Profile and X-WAP-Profile-Diff headers [UAPROF] MAY be included in the OPTIONS response. The profile referenced by these headers should be resolved as per the resolution rules specified in section 6.4 of [UAProf]. If, in the resolved profile, a push component exists its attributes should used when establishing the clients CPI. However attributes in the resolved profile MUST NOT supersede the specific CPI headers, defined in section 7.2.5.5, where available (as per section 8.1.2.3 of [UAProf]).

The headers (and associated attribute values) used to convey user agent profile information [UAProf] MUST NOT be used in the calculation of client's CPITag value, as defined in section 7.2.5.1.

7.2.6 Mutual Terminal/PPG Identification and Authentication

When an active TCP connection has been established (see section 7.2.4.1), the PPG SHOULD identify the terminal to ensure that pushed content is forwarded to the intended terminal. The terminal can also be authenticated if requested by the PPG.

A PPG uses a terminal's Terminal-ID to uniquely identify that terminal. The means for conveying the Terminal-ID between the terminal and the PPG are described in the subsequent sections, and it is formatted according to the following rules:

- If the terminal supports WAP Provisioning [ProvArch] it MUST use the value of the `PXAUTH-ID` parameter [ProvCont] (or the fallback value if the parameter is missing) if it is able to select the appropriate `PXLOGICAL` in the configuration context used.
- If the terminal does not support WAP Provisioning, or if it fails to select the appropriate `PXLOGICAL`, the Terminal-ID MUST be formatted in accordance with [ClientID].

Similarly the terminal SHOULD identify the PPG to ensure that content from non-desirable PPGs can be rejected.

7.2.6.1 Un-authenticated Identification

The terminal MUST include its Terminal-ID in the response to the OPTIONS request (see section 7.2.5.1) using the `X-Wap-Terminal-Id` header (see section 7.2.6.1.1), unless:

- authentication has been requested by the PPG using the `X-Wap-Authenticate` header, implying that the Terminal-ID will be conveyed as part of the `X-Wap-Authorization` header (as described in section 7.2.6.2.2), or
- the terminal requests the PPG to authenticate itself as defined in section 7.2.6.2

For PPG identification purposes the terminal MAY use the remote address of the active TCP connection.

7.2.6.1.1 X-Wap-Terminal-Id Header

This general header is used to carry the terminal's Terminal-ID. The ABNF [RFC2234] format of this header is:

```
X-Wap-Terminal-Id = "X-Wap-Terminal-Id" ":" Terminal-ID
; Terminal-ID, a terminal identifier that MUST be formatted
; as defined in section 7.2.6
```

7.2.6.2 Authenticated Identification

The authentication schemes described in this section allow the PPG to be authenticated by the terminal and vice versa. The terminal (acting as an HTTP server) uses the mechanisms defined in [RFC2617] to authenticate the PPG (acting as an HTTP client), while similar methods are defined by this specification to allow the PPG to authenticate the terminal (RFC2617 only specifies how an HTTP client can be authenticated by an HTTP server).

7.2.6.2.1 PPG Authentication

For PPG authentication purposes, both the terminal and the PPG MUST support the "basic" authentication scheme, and MAY support the "digest" authentication scheme, as defined in [RFC2617].

The restrictions defined in section 0 apply to the use of the `WWW-Authenticate` header, with the following exception:

- `realm` MUST be the appropriate Terminal-ID value of the terminal requesting authentication
- `domain` MUST be `/wappush`

Further, the restrictions defined in section 7.2.6.2.2.2 apply to the use of the `Authorization` header with the following exceptions:

- `digest-URI` and `Method` are used as specified in [RFC2617]
- `username` MUST include the identity of the PPG, formatted as the `PX-LOGICAL.PROXY-ID` parameter defined in [ProvCont]

The terminal MAY use status code 401 "Unauthorized" to request the PPG to supply, or re-send, its authorisation credentials.

7.2.6.2.2 Terminal Authentication

Terminal authentication is achieved by a mechanism similar to that described in [RFC2617], but modified so it can be used to authenticate an HTTP server instead of an HTTP client, and just like [RFC2617] it provides a means for both "basic" and "digest" authentication. Both the terminal and the PPG MUST support the "basic" authentication scheme, and MAY support the "digest" authentication scheme.

Challenges and credentials are carried between the PPG and the terminal using the following two HTTP headers (defined in subsequent sections):

- `X-Wap-Authenticate`: used by the PPG to request terminal authentication and carry the challenge

- X-Wap-Authorization: used by the terminal to carry its credentials.

To request authentication from the terminal, the PPG includes the X-Wap-Authenticate header in a request sent to the terminal. The terminal responds with its authorization credentials in the X-Wap-Authorization header if it accepts the challenge. The terminal MUST NOT include the X-Wap-Authorization header in a response unless the X-Wap-Authenticate header was present in the corresponding request.

If the terminal does not accept the challenge sent by the PPG it MUST respond with status code 412 "Precondition Failed" and include an auth-param directive [RFC2617] in the X-Wap-Authorization header with the following ABNF [ABNF] definition:

```
x-wap-auth-status = "x-wap-auth-status" "=" x-wap-auth-status-value

x-wap-auth-status-value = "failed_retry" | "failed_noretry"
```

The token "failed_retry" indicates that the PPG MAY retry the request by sending the X-Wap-Authenticate header anew. If the field is set to "failed_noretry", the PPG MUST NOT re-send the X-Wap-Authenticate header.

If the PPG does not accept the credentials supplied by the terminal it MUST re-send the request and include the X-Wap-Authenticate header with the x-wap-auth-status field value set to the token "failed_retry" or "failed_noretry". The token "failed_retry" indicates that the terminal MUST either retry to authenticate itself by re-sending the X-Wap-Authorization header or terminate the connection with the PPG. If the field is set to "failed_noretry", the terminal MUST terminate the connection.

7.2.6.2.2.1. X-Wap-Authenticate Header

The PPG uses the X-Wap-Authenticate header to request authentication from a terminal and carry the challenge. The semantics of this header are as defined in [RFC2616] & [RFC2617] for the WWW-Authenticate header except that it is included in requests instead of responses. The following restrictions apply:

- realm MUST include the identity of the PPG, formatted as the PX-LOGICAL.PROXY-ID parameter defined in [ProvCont]
- domain MUST NOT be used
- stale MUST NOT be used
- algorithm MUST NOT be used
- qop-options MUST NOT be used
- algorithm MUST be "SHA-1"

The nonce parameter should be uniquely generated each time the X-Wap-Authenticate header is sent.

7.2.6.2.2.2. X-Wap-Authorization Header

The terminal uses the X-Wap-Authorization header to carry the authentication response back to the PPG. The semantics of this header is as defined in [RFC2616] and [RFC2617] for the Authorization header except that it is included in responses instead of requests. The following restrictions apply:

- username MUST include the appropriate Terminal-ID value of the terminal being authenticated
- digest-URI MUST be /wappush
- algorithm MUST NOT be used
- cnonce MUST NOT be used

- `message-qop` MUST NOT be used
- `nonce-count` MUST NOT be used
- `algorithm` MUST be "SHA-1"

The computing of the `request-digest` value is done as defined in [RFC2617] except for the following:

- SHA-1 MUST be the hash algorithm used
- `auth` MUST be the quality of protection used
- `Method` in A2 MUST be the method used in the request containing the `X-Wap-Authenticate` header
- `digest-uri-value` in A2 MUST be the `request-uri` [RFC2616] (in this case `/wappush`) used in the request containing the `X-Wap-Authenticate` header

7.2.6.2.3 Examples

A PPG desiring to authenticate a terminal using "digest" authentication, when an active TCP connection has been established, sends the `X-Wap-Authenticate` header in the OPTIONS command request and analyses the terminal credentials part of the `X-Wap-Authorization` header in the OPTIONS command response.

A terminal desiring to authenticate a PPG responds with status code 401 and includes the `WWW-Authenticate` header in the OPTIONS command response and analyses the PPG credentials part of the `Authorization` header, when/if the PPG sends the request anew, before sending the response to the last OPTIONS request.

The figure below illustrates the procedure when the terminal accepts an unauthenticated registration request:

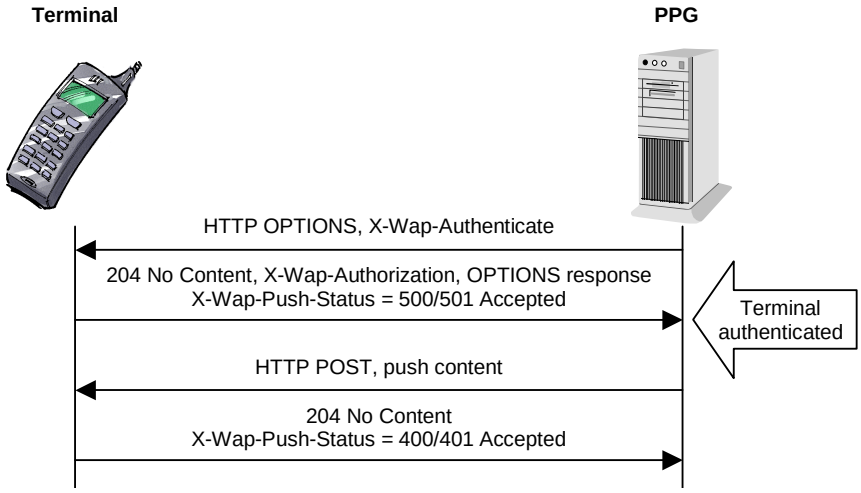

Figure 9: Terminal accepts unauthenticated registration request

As described above, the terminal may choose to request the PPG to be authenticated before accepting the registration request. The procedure would then be as illustrated in the figure below:

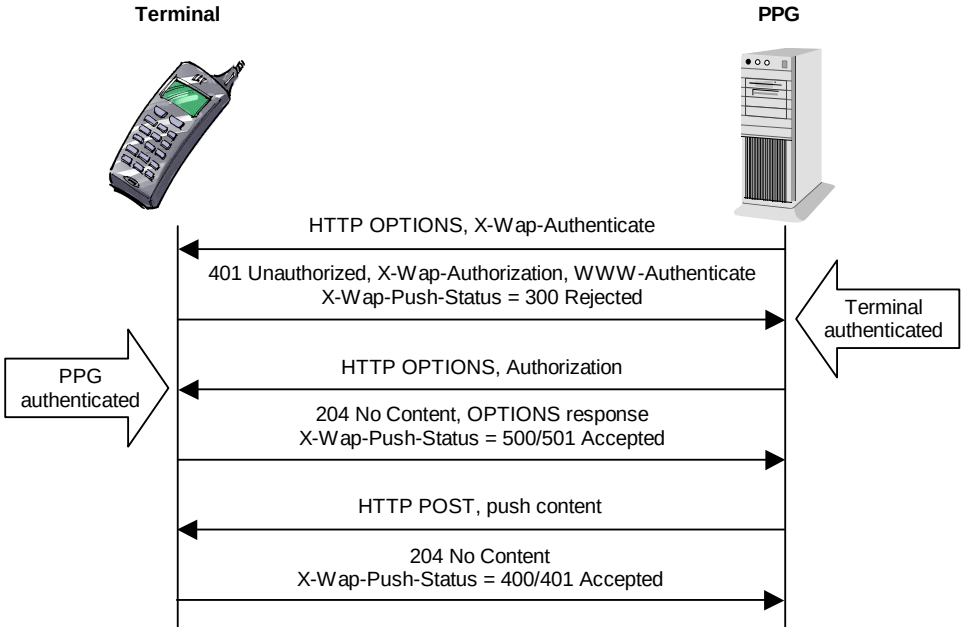

Figure 10: Terminal requests PPG authentication prior registration

In the case of "basic" authentication, the same procedure can be used. It is then however also possible for the PPG to include the `Authorization` header without first receiving the `WWW-Authenticate` header (if the PPG includes it in the initial OPTIONS, the extra roundtrip where the terminal requests the PPG to authenticate itself can be avoided) as illustrated in the figure below.

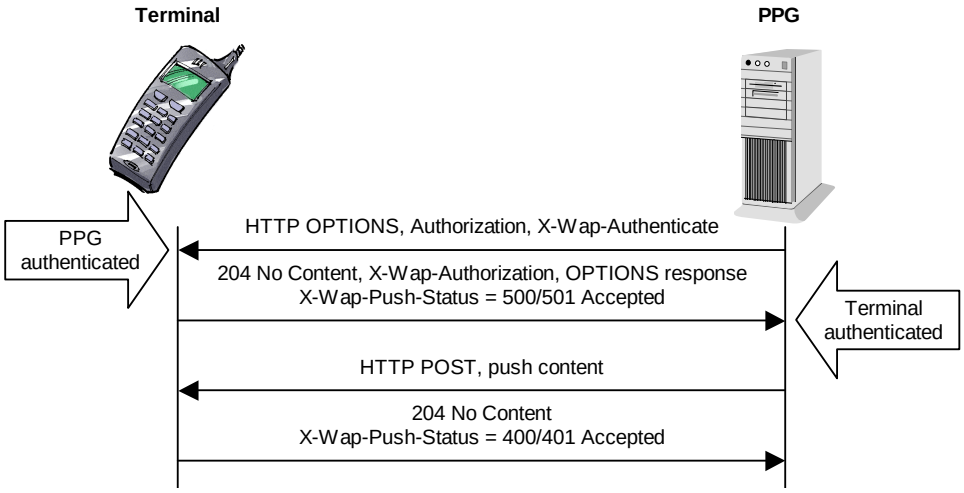

Figure 11: Use of "Basic" authentication

The authentication scheme is of course not restricted to be used only with the first HTTP requests sent to the terminal when an active TCP connection has been established, it may be used at any time and with any method if desired.

7.3 Application Addressing

The PPG MUST address the terminal push application for any push request using the `/wappush` abs_path as the URI of the POST request (see section 7.4).

The terminal MUST use the `X-Wap-Application-Id` value to route the push request to the intended application. When no `X-Wap-Application-Id` header is provided, the terminal MUST assume that the intended application is the WML user agent.

7.4 Content Push

Push messages are delivered to the terminal using the HTTP POST method [RFC2616]. This section defines the format for the POST request and its corresponding response.

7.4.1 POST Request Format

The message body of the POST request, using `/wappush` as Request-URI and an empty `HOST` header field, carries the content and headers to be delivered to the addressed application (see section 7.3 for details about application addressing) enclosed in an application/http response entity body [RFC2616]. The entity headers that may be used in the application/http entity body are defined in [PushMsg]. These headers are delivered end-to-end, i.e. from PI to terminal. The status-line in the application/http entity body contains a status code legal for an HTTP response. The `X-Wap-Push-ProvURL` header MAY be included in the request. See section 7.2.5.4 for further details.

Request headers, besides those specified in this specification, are defined in [RFC2616].

The `X-Wap-Push-Info` header MAY be included in the POST request to convey push specific information to the terminal. It is described in the section below.

7.4.1.1 The X-Wap-Push-Info Header

The `X-Wap-Push-Info` header is a request header used in a POST request sent by the PPG to provide the terminal with the following indications regarding each particular push transaction. It can carry the following attribute tokens:

- `authenticated`: used as the *Authenticated Flag* described in section 6.2.4. The *Initiator URI* mentioned in that section is represented by the `X-Wap-Initiator-URI` defined in [PushMsg].
- `trusted`: used as the *Trusted Flag* as described in section 6.2.5.
- `last`: used as the *Last Flag* as described in section 6.2.6.
- `response`: indicates that a message body MAY be included in the response to the POST request. The terminal MUST NOT include any message body in the response if this token is not present.

The ABNF [RFC2234] format is:

```
X-Wap-Push-Info = "X-Wap-Push-Info" ":" token *("," token)
token = ("authenticated" | "trusted" | "last" | "response")
```

Unrecognised token values MUST be ignored by the terminal.

7.4.2 POST Response Format

The response to the HTTP POST method MUST contain a status line reflecting the outcome of the request. Status codes 200 and 204 are equivalent in the respect that they both indicate that the addressed terminal application has accepted the push content for processing. Status code 204 "No Content" is used if the response does not contain a message body, and status code 200 "OK" is used if a message body is included (the response MAY contain a message body if the PPG explicitly permits it in the corresponding request as described in section 7.4.1.1 – however, neither the contents, nor the use, of the message body is specified by this specification).

Other allowed status codes, reflecting the outcome of the HTTP POST request, are defined in [RFC2616]. The `X-Wap-Push-Status` header (see section 7.4.2.1) MUST be included in the response to reflect the outcome of the push submission.

Response headers, besides those specified in this specification, are defined in [RFC2616].

7.4.2.1 The X-Wap-Push-Status Header

The `X-Wap-Push-Status` header is used to indicate the outcome of a registration request or a push request, i.e. it is used to convey statuses not pertaining to HTTP. The header MUST be included in responses to all registration and push requests. The ABNF [RFC2234] format is:

```
X-Wap-Push-Status = "X-Wap-Push-Status" ":" Status-Line
Status-Line = Status-Code [SP Reason-Phrase]
Status-Code = 3DIGIT
Reason-Phrase = *VCHAR
; Status-Code values are defined in the table below
; Reason-Phrase is an appropriate textual phrase (optional)
; Example: X-Wap-Push-Status: 237 Resource Shortage
; Status-Code 234-299: Push request rejected
; Status-Code 300-399: Registration request rejected
; Status-Code 400-499: Push request accepted
; Status-Code 500-599: Registration request accepted
; Status-Code 600-699: General rejection reasons
```

The following `Status-Code` values are allowed in the `X-Wap-Push-Status` header:

Status Code	HTTP Method	Retries Allowed[1]	Description
234	POST	Yes	Push request rejected, see USERREQ in section 6.1.3.3
235	POST	No	Push request rejected, see USERRFS in section 6.1.3.3
236	POST	No	Push request rejected, see USERPND in section 6.1.3.3
237	POST	Yes	Push request rejected, see USERDCR in section 6.1.3.3
238	POST	No	Push request rejected, see USERDCU in section 6.1.3.3
256	POST	No	Push request rejected, CPITag not present or mismatching
257	POST	No	Push request rejected, matching provisioning context not found
300	OPTIONS	Yes	Registration request rejected, retries allowed
301	OPTIONS	No	Registration request rejected, no retries
302	OPTIONS	No	Registration request rejected, matching provisioning context not found
400	POST	N/A	Push request accepted, CPITag not present or matching
401	POST	N/A	Push request accepted, CPITag mismatch
500	OPTIONS	N/A	Registration request accepted, CPITag matching
501	OPTIONS	N/A	Registration request accepted, CPITag not present or mismatching
600	*[2]	N/A	Request rejected, the terminal does not support the OTA-HTTP version indicated by the PPG

[1] *Indicates if the PPG may re-send the request without changes*
[2] *Any method*

7.4.3 Example

Below is an example of a push request containing a Service Indication that allows the user to invoke his/her email service:

```
POST /wappush HTTP/1.1
Host:
Date: Tue, 31 Jul 2001 10:13:05 GMT
Content-Type: application/http
Content-Length: 504
X-Wap-Push-OTA-Version: 1.0

HTTP/1.1 200 OK
Date: Tue, 31 Jul 2001 10:13:00 GMT
Last-modified: Tue, 31 Jul 2001 10:13:00 GMT
Content-Language: en
Content-Length: 268
Content-Type: text/vnd.wap.si
X-Wap-Application-Id: x-wap-application:wml.ua

<?xml version="1.0"?>
     <!DOCTYPE si PUBLIC "-//WAPFORUM//DTD SI 1.0//EN"
                   "http://www.wapforum.org/DTD/si.dtd">
<si>
    <indication      href="http://www.xyz.com/email/123/abc.wml"
                                  created="2001-07-31T10:13:00Z"
                                  si-expires="2001-08-07T10:13:00Z">You have 4
new emails</indication>
</si>
```

If the terminal accepts the request, the response would look like:

```
HTTP/1.1 204 No Content
X-Wap-Push-Status: 400 Accepted
```

7.5 Version Control

The OTA protocol over HTTP provides a simple mechanism for protocol version discovery by using a <major>.<minor> numbering scheme. The <major> and <minor> numbers should be interpreted as integer values, implying that e.g. 3.5 is a lower version than 3.12. A star (*) MAY be used as <minor> version to indicate support (or acceptance) for all <minor> versions for a given <major> version.

The <minor> number is incremented when the newer version still can be used to communicate with a party (PPG/terminal) supporting a lower <minor> version with identical <major> version, although optional features might not work.

The <major> version is incremented when the protocol is changed in a manner that the new version cannot be used with the current version. Two parties should not expect to be able to communicate using protocols with different <major> version numbers.

The version numbers supported are conveyed using the `X-Wap-Push-OTA-Version` header. The ABNF [RFC2234] format is:

```
X-Wap-Push-OTA-Version: "X-Wap-Push-OTA-Version " ":" supported-versions
supported-versions = version-number *("," version-number)
version-number = *DIGIT "." ((*DIGIT) / "*")
; Example:  X-Wap-Push-OTA-Version:1.0,1.3,2.*,3.4
```

The version numbers are listed in order of preference, with the most preferred first.

The `X-Wap-Push-OTA-Version` header MUST be included in an HTTP response if it was included in the corresponding request. The header MUST be present in the first HTTP request sent over an active TCP connection, and MAY be present in subsequent requests.

If the terminal is not willing to accept any of the versions indicated by the PPG, the terminal MUST include the `X-Wap-Push-Status` header with the value 600 and an appropriate textual message (e.g. "Version Not Supported") in the response.

The version specified by this specification is 1.0.

7.6 Bearer Indication

The terminal might choose to register with a PPG using different bearers. For example, the SIR mechanism provides a means for the PPG to advertise different desired bearers to be used by the terminal when establishing IP connectivity with the network (see section 8.1).

The registration mechanism provides a means for the client to report which bearer it used when it established IP connectivity as described below. This information MAY be used by the PPG to perform bearer selection (e.g. delivery of some bulky content might not be feasible over the most constrained bearers).

The terminal MUST indicate the bearer used during registration by including the `X-Wap-Bearer-Indication` header in the response to the OPTIONS method (see section 7.2.5.1).

7.6.1 X-Wap-Bearer-Indication Header

The terminal uses the `X-Wap-Bearer-Indication` header to indicate the bearer used for a particular registration. The ABNF [RFC2234] format of this header is:

```
X-Wap-Bearer-Indication = "X-Wap-Bearer-Indication" ":" bearer-type
Bearer-type = 2HEXDIG
; Bearer-type as defined in WDP
```

8. Session Initiation Request

Since content push is asynchronous by nature, it is possible that no push session exists (OTA-WSP), that no active TCP connection has been established (OTA-HTTP), or that the desired bearer is not utilized when content is about to be pushed from the PPG to the terminal. The *Session Initiation Application* (SIA) residing in the terminal allows a PPG to request a terminal to establish a push session or an active TCP connection, using a specific bearer. The process of sending SIA content to a mobile terminal is referred to as *Session Initiation Request* (SIR) independent of the protocol variant to be used. An SIR can be delivered using either connectionless or connection-oriented push.

The SIA content type contains separate lists of contact points for OTA-WSP and OTA-HTTP. This implies that the originator of an SIR (usually a PPG) can choose to indicate either of them or both. The two lists of contact points are to be considered as alternatives, and hence the terminal MUST only use one of them (that is, one protocol variant).

If contact points are included for only one protocol variant, and the terminal does not support that variant, the terminal MUST NOT attempt to use another protocol variant when contacting any of those contact points.

If an SIR contains lists of contact points for both OTA-WSP and OTA-HTTP, it is left to the terminal's discretion to decide which protocol variant it shall use.

The subsequent sections describe how a SIR is carried out in OTA-HTTP and in OTA-WSP respectively.

8.1 SIR in OTA-HTTP

8.1.1 Session Initiation Application

SIA MUST be supported both by terminals and PPGs implementing OTA-HTTP.

8.1.2 PPG Procedure

A PPG can instruct a terminal to establish an active TCP connection by sending an SIR to the SIA in the terminal, indicating contact points for OTA-HTTP. The SIA is addressed by its registered Application-ID [OMNA].

8.1.3 Terminal Procedure

When/if acting upon an SIR, the terminal MUST take the following actions:

- Establish IP connectivity with the network, if not already done
- Proceed with the TO-TCP connection procedure described in section 7.2.4.1.1

If multiple contact points (one or more PPGs) are included in the SIR, the terminal SHOULD establish active TCP connections towards each of those contact points.

If the terminal supports OTA-HTTP-TLS it MUST ensure that a TLS session is established on the active TCP connection it creates towards the PPG, if the secure transport service is requested (by indicating the secure registered port, or a provisioned port known to support TLS, in the SIR).

8.2 SIR in OTA-WSP

8.2.1 Session Initiation Application

SIA MUST be supported both by a terminal and a PPG implementing connection-oriented push using OTA-WSP.

8.2.2 PPG Procedure

A PPG can instruct a terminal to establish one or more push sessions by sending an SIR to the SIA in the client, indicating contact points for OTA-WSP. The SIA is addressed by its registered Application-ID [OMNA].

8.2.3 Terminal Procedure

When/if acting upon an SIR, the terminal MUST take the following actions:

- Establish connectivity with the network, if not already done
- Establish push sessions towards the contact points in the SIR

If multiple contact points (one or more PPGs) are included in the SIR, the terminal SHOULD establish push sessions towards each contact point indicating its subset of supported Application-IDs specified in the SIR. However, the terminal MAY indicate (e.g. due to privacy concerns) that it accepts any Application-ID. It is the responsibility of the client to clean up the stale push sessions, if any.

The terminal MUST ensure that a WTLS secure connection exists before it creates the new push session, if the secure transport service is requested (by indicating the secure registered port, or a provisioned port known to support WTLS, in the SIR).

8.3 Security Considerations

To protect against denial of service attacks, the terminal SHOULD implement a lockout timer. If the terminal receives any additional SIRs during the lockout interval, it should defer processing or discard them until the timer expires. If the requested push session(s) is successfully established (OTA-WSP), or if the active TCP connection(s) is successfully established (OTA-HTTP), the lockout timer SHOULD be reset. The value of the lockout timer interval is implementation specific.

To protect against spoofing, the terminal SHOULD validate the SIR by comparing the source address of the PDU that carries the SIA content with a pre-existing list of authorised PPGs. The SIR SHOULD be ignored if the validation fails.

The above measures are applicable if the SIR is received on a non-secure port. If a secure port is used, these measures are generally not necessary.

8.4 SIA Content Based Protocol Data Unit

The content type, `application/vnd.wap.sia`, is defined and encoded as follows:

	Field Name	Type
Common fields	Version	uint8*
	AppIdListLen	uintvar*; number of octets* for Application-ID List field
	Application-ID List	AppIdListLen octets
OTA-WSP specific fields	ContactPointsLen WSP	uintvar; number of octets for Contact Points WSP field
	Contact Points WSP	ContactPointsLen octets
Fields for other protocols	ContactPointsLen	uintvar; number of octets for Contact Points field
	Contact Points	ContactPointsLen octets
	ProtOptsLen	uintvar; number of uintvar encoded octets for ProtOpts field
	ProtOpts	ProtOptsLen uintvar
ProvURL fields	ProvURLLen	uintvar; number of octets for ProvURLfield
	ProvURL	ProvURLLen octets
CPITag fields	CPITagLen	uintvar; number of CPITags (each 4*octet) in the CPITag field
	CPITag	CPITagLen 4*octet

** As defined in [WSP]*

The ***Version*** field indicates the version of SIA content type. For this specification version, its value is 1. Future versions of SIA should only add new fields at the end of this content type, if such are needed, to ensure maximum backward compatibility. A terminal MUST accept version numbers higher than 1, and ignore unknown fields (i.e. fields included in later versions). To ensure that a terminal implementing connection-oriented push using OTA-WSP will be compatible with older PPGs (using version 0), such terminals MUST also support SIA version 0. Version 0 and version 1 are identical with respect to the common and WSP specific fields, except for the version number.

AppIdListLen, ***ContactPointsLen WSP***, ***ContactPointsLen, ProtOptsLen, ProvURLLen*** and ***CPITagLen*** indicate the length of the following field (a length of zero is allowed). Each length is encoded using the variable-length *uintvar* integer format.

The ***Application-ID List*** field contains a list of Application-IDs to which the PPG wishes to send Push messages. The terminal, in turn, indicates the subset of supported Application-IDs when a push session is established (OTA-WSP), or when a registration takes place (OTA-HTTP), by sending `accept-application` headers [WSP]. See sections 8.1.3 and 8.2.3 for details on how to use this field when multiple contact points are specified.

The ***Contact Points WSP*** field contains a list of server addresses the client should contact to establish a WSP push session (OTA-WSP). Each address in the field uses the *AddressType* as defined in [WSP].

The ***Contact Points*** field contains a list of PPG addresses the terminal should contact using another protocol than OTA-WSP (currently only OTA-HTTP). In the case of OTA-HTTP, the terminal should establish an active TCP connection (or connections) when contacting the PPG(s) using TO-TCP. Each address in the field uses the *AddressType* as defined in [WSP].

The ***ProtOpts*** field contains a list of identifiers (each represented using a binary representation of its decimal value, encoded as uintvar) that identify the protocol, and its associated options, to be used when the terminal contacts the contact points specified in the *Contact Points* field. The first identifier identifies the protocol to be used when contacting the first contact point, the second identifier identifies the protocol to be used when contacting the second contact point, and so on. If the number of listed protocol identifiers does not match the number of contact points specified in the *Contact Points* field, the first protocol identifier MUST be used for all contact points. If the *Protocol* field is empty, or omitted, the default protocol identifier is 0 (zero). If the terminal receives an unknown identifier it MUST NOT attempt to contact the associated contact point(s).

Allowed protocol identifiers are:

Identifier	Description
0	OTA-HTTP, no CPITag present
1	OTA-HTTP, CPITag present

OMA reserves identifiers zero through 255 for internal use, while identifiers 256 through 16383 are available for private assignment through OMNA [OMNA].

The ***ProvURL*** field contains the `ProvURL` [ProvCont] parameter value assigned to the configuration context [ProvCont] that the terminal should use when contacting the contact point(s) listed in the SIR. The following rules apply:

- If the terminal supports WAP Provisioning:
  - If the ProvURL field is non-empty and it matches one of the terminal's configuration contexts, the matching configuration context MUST be used.
  - If the ProvURL field is empty, it is left to the discretion of the terminal to select the appropriate configuration context among those having an empty ProvURL.
  - If either action listed in the two above bullets fails, it is left to the discretion of the terminal if and how to contact the contact points specified in the SIR.
- If the terminal does not support WAP Provisioning, this field can be ignored.

If the ProvURLLen field indicates lengths one through four octets, the value of the ProvURL field MUST contain a truncated hash of the ProvURL calculated using SHA-1 [SHA]. A ProvURLLen value of one indicates that the first byte of the output is used, a ProvURLLen of two indicates that the first two bytes of the output is used, and so on. ProvURLLen values above four indicate that the ProvURL is represented in its full textual representation (ASCII encoded).

The ***CPITag*** field is used to convey a list of CPITags assumed to be valid by the PPG. Each CPITag is represented by the 4 octets (non-encoded, i.e. not encoded using base64) previously sent from the terminal to the PPG in the `X-Wap-CPITag` header (see section 7.2.5.3). The first element in the list of CPITags is interlinked with the first contact point specified in the *Contact Points* field for which the *ProtOpts* identifier indicates that the CPITag is present, the second element in the list of CPITags is interlinked with the second contact point for which the *ProtOpts* identifier indicatesthat the CPITag is present, and so on. If the number of listed CPITags does not match the number of contact points specified in the *Contact Points* field, for which the *ProtOpts* identifier indicates that the CPITag is present, the first CPITag MUST be used for all those contact points. If a *ProtOpts* identifier indicates that the CPITag is present, but the *CPITag* field is empty, the terminal MUST handle the SIR as if the CPI is not known by the PPG to provide a reasonable level of tolerance towards errors in the content.

Unused fields may be omitted only if other fields do not follow them, implying that a terminal MUST accept truncated SIRs. This means, for example, that if the PPG does not wish to indicate an OTA-WSP Contact Point, the *ContactPointsLen WSP* field MUST be present with a value of 0. On the other hand, if the PPG wishes to only indicate an OTA-WSP contact point, the fields following the WSP specific fields may be omitted

Appendix A. Static Conformance Requirements (Normative)

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

A.1 Client/Terminal Features

Item	Function	Reference	Status	Requirement
OTA-CL-C-001	Connectionless Push	5, 6.2.1	M	WSP-CL-C-002 AND WSP-CL-C-003 AND WSP-CL-C-020 AND WDP:MCF
OTA-CL-C-002	Non-secure Port for connectionless push	6.2.1	M	WDP-RP-C-001
OTA-CL-C-003	Secure Port for WTLS for connectionless push	6.2.1	O	OTA-CL-C-001 AND WDP-RP-C-002 AND WTLS:MCF AND WTLS:WTLS-C-007
OTA-CO-C-001	Connection-oriented push	5	O	OTA-CO-C-002 OR OTA-CO-C-003
OTA-CO-C-002	Connection-Oriented Push using OTA-WSP	6.2.2	O	(OTA-WSP-C-001 OR OTA-WSP-C-002) AND (OTA-WSP-C-003 OR OTA-WSP-C-004) AND OTA-WSP-C-005
OTA-WSP-C-001	Connection-oriented Confirmed Push	6.2.2	O	WSP-CO-C-001 AND WSP-CO-C-011 AND WSP-CO-C-034
OTA-WSP-C-002	Connection-oriented Unconfirmed Push	6.2.2	O	WSP-CO-C-001 AND WSP-CO-C-010
OTA-WSP-C-003	Use non-secure transport service	6.2.2	O	
OTA-WSP-C-004	Use secure transport service with WTLS	6.2.2	O	WTLS:MCF AND WTLS: WTLS-C-007
OTA-WSP-C-005	SIA/SIR	8 8.2 8.4	O	
OTA-WSP-C-006	Application Addressing	6.2.3	M	
OTA-WSP-C-007	Application Dispatching	6.3.1	M	
OTA-WSP-C-008	Initiator Authentication	6.2.4	O	
OTA-WSP-C-009	Bearer Selection	6.2.6	O	
OTA-WSP-C-010	Bearer Control	6.2.6	O	
OTA-WSP-C-011	Security Considerations	8.3	O	

OTA-CO-C-003	Connection-Oriented Push using OTA-HTTP	7	O	OTA-HTTP-C-001 AND OTA-HTTP-C-002 AND OTA-HTTP-C-005 AND OTA-HTTP-C-006 AND OTA-HTTP-C-007 AND OTA-HTTP-C-009 AND OTA-HTTP-C-010 AND OTA-HTTP-C-011 AND OTA-HTTP-C-012 AND OTA-HTTP-C-013 AND OTA-HTTP-C-015 AND OTA-HTTP-C-016 AND HTTP-C-S-001
OTA-HTTP-C-001	TO-TCP	7.2.4.1	O	TCP:MCF
OTA-HTTP-C-002	PO-TCP	7.2.4.1	O	TCP:MSF
OTA-HTTP-C-003	Secure (OTA-HTTP-TLS) TO-TCP	7.2.4.1	O	TLS:MCF
OTA-HTTP-C-004	Secure (OTA-HTTP-TLS) PO-TCP	7.2.4.1	O	TLS:MCF
OTA-HTTP-C-005	Registration	7.2.5 7.2.5.1	O	
OTA-HTTP-C-006	Registration Validation	7.2.5 7.2.5.2	O	
OTA-HTTP-C-007	Support Un-authenticated Terminal Identification	7.2.6 7.2.6.1	O	
OTA-HTTP-C-008	Support Un-authenticated PPG Identification	7.2.6 7.2.6.1	O	
OTA-HTTP-C-009	Support Authenticated Terminal Identification	7.2.6 7.2.6.2	O	
OTA-HTTP-C-010	Support Authenticated PPG Identification	7.2.6 7.2.6.2	O	
OTA-HTTP-C-011	Application Addressing	7.3	O	
OTA-HTTP-C-012	Content Push	7.4	O	
OTA-HTTP-C-013	Version Control	7.5	O	
OTA-HTTP-C-014	Security Considerations	8.3	O	
OTA-HTTP-C-015	Bearer Indication	7.6	O	
OTA-HTTP-C-016	SIA/SIR	8 8.1 8.4	O	
OTA-HTTP-C-017	Support for the `X-Wap-Push-ProvURL` header	7.2.5.4	O	ProvCont-CB-C-001

A.2 Server/PPG Features

Item	Function	Reference	Status	Requirement
OTA-CL-S-001	Connectionless Push	5, 6.2.1	M	WSP-CL-S-002 AND WSP-CL-S-003 AND WSP-CL-S-020 AND WDP:MCF
OTA-CO-S-001	Connection-oriented push	5	O	OTA-CO-S-002 OR OTA-CO-S-003
OTA-CO-S-002	Connection-Oriented Push using OTA-WSP	6.2.2	O	(OTA-WSP-S-001 OR OTA-WSP-S-002) AND (OTA-WSP-S-003 OR OTA-WSP-S-004) AND OTA-WSP-S-005
OTA-WSP-S-001	Connection-oriented Confirmed Push	6.2.2	O	WSP-CO-S-001 AND WSP-CO-S-011
OTA-WSP-S-002	Connection-oriented Unconfirmed Push	6.2.2	O	WSP-CO-S-001 AND WSP-CO-S-010
OTA-WSP-S-003	Use non-secure transport service	6.2.2	O	WDP-RP-S-004
OTA-WSP-S-004	Use secure transport service with WTLS	6.2.2	O	WDP-RP-S-006 AND WTLS:MCF
OTA-WSP-S-005	SIA/SIR	8 8.2 8.4	O	
OTA-WSP-S-006	Application Addressing	6.2.3	M	
OTA-WSP-S-007	Initiator Authentication	6.2.4	O	
OTA-WSP-S-008	Bearer Selection	6.2.6	O	
OTA-WSP-S-009	Bearer Control	6.2.6	O	
OTA-CO-S-003	Connection-Oriented Push using OTA-HTTP	7	O	OTA-HTTP-S-001 AND OTA-HTTP-S-007 AND OTA-HTTP-S-008 AND OTA-HTTP-S-009 AND OTA-HTTP-S-010 AND OTA-HTTP-S-011 AND OTA-HTTP-S-012 AND OTA-HTTP-S-014 AND OTA-HTTP-S-015 AND HTTP-S-C-001
OTA-HTTP-S-001	TO-TCP	7.2.4.1	O	TCP:MSF
OTA-HTTP-S-002	PO-TCP	7.2.4.1	O	TCP:MCF
OTA-HTTP-S-003	Secure (OTA-HTTP-TLS) TO-TCP	7.2.4.1	O	TLS:MSF
OTA-HTTP-S-004	Secure (OTA-HTTP-TLS) PO-TCP	7.2.4.1	O	TLS:MSF
OTA-HTTP-S-005	Registration	7.2.5 7.2.5.1	O	
OTA-HTTP-S-006	Registration validation	7.2.5 7.2.5.2	O	
OTA-HTTP-S-007	Support Un-authenticated Terminal Identification	7.2.6.1	O	

OTA-HTTP-S-008	Support Authenticated Terminal Identification	7.2.6.2	O	
OTA-HTTP-S-009	Support Authenticated PPG Identification	7.2.6.2	O	
OTA-HTTP-S-010	Application Addressing	7.3	O	
OTA-HTTP-S-011	Content Push	7.4	O	
OTA-HTTP-S-012	Version Control	7.5	O	
OTA-HTTP-S-013	Security Considerations	8.3	O	
OTA-HTTP-S-014	Bearer Indication	7.6	O	
OTA-HTTP-S-015	SIA/SIR	8 8.1 8.4	O	
OTA-HTTP-S-016	Support for the `X-Wap-Push-ProvURL` header	7.2.5.4	O	

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
WAP-189-PushOTA	17 Feb 2000	WAP 1 Conformance Release Push Over The Air Specification
WAP-189_100-PushOTA	11 Dec 2000	Approved SIN on WAP 1 Conformance Release
WAP-189_101-PushOTA	26 Sep 2001	Approved SIN on WAP 1 Conformance Release
WAP-235-PushOTA	25 Apr 2001	WAP 2 Conformance Release Push Over The Air Specification
OMA-WAP-TS-PushOTA-V2_1-20110405-A	05 Apr 2011	Status changed to Approved by TP TP ref # OMA-TP-2011-0098-INP_Push_V2_1_ERP_for_Final_Approval