Push Over The Air

Candidate Version 2.3 – 16 Mar 2010

Open Mobile Alliance

OMA-TS-PushOTA-V2_3-20100316-C

Contents

Figures

1. Scope

This specification defines the Over the Air protocol for delivery of content to a 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 be used over the WSP protocol [W-TCP], HTTP 1.1 [RFC2616], SIP [SIPPush], or point-to-multipoint bearers such as Cell Broadcast (CBS) [TS23.041] and 3GPP MBMS [TS26.346].

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.
- means for bearer selection and control.
- means for authentication.

2. References

2.1 Normative References

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

[BCAST-Security] “Service and Content Protection for Mobile Broadcast Services”. Open Mobile Alliance™ OMA-TS-BCAST_SvcCntProtection-V1_0. URL:http://www.openmobilealliance.org

[BCAST-SvcGuide] “Service Guide for Mobile Broadcast Services”. Open Mobile Alliance™. OMA-TS-BCAST_ServiceGuide-V1_0. URL:http://www.openmobilealliance.org

[BCAST10-XMLSchema-FD-FDT] "Mobile Broadcast Services – XML Schema for File Description Table", Open Mobile Alliance™, OMA-SUP-XSD_bcast_fd_fdt-V1_0, URL: http://www.openmobilealliance.org/

[DMSTDOBJ] "OMA Device Management Standardized Objects, Version 1.2". Open Mobile Alliance™ OMA-TS-DM-StdObj-V1_2. URL:http://www.openmobilealliance.org

[DM-TND-V1-2] "OMA Device Management Tree and Description, Version 1.2". Open Mobile Alliance™ OMA-TS-DM_TND-V1_2 URL:http://www.openmobilealliance.org

[PROVBOOT] "Provisioning Bootstrap 1.1". Open Mobile Alliance™. OMA-WAP-ProvBoot-v1_1. URL:http://www.openmobilealliance.org

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

[PushArch] “Push Architecture”. Open Mobile Alliance™. OMA-AD-Push-V2_3. URL:http://www.openmobilealliance.org/

[PushCAI] "Push Client - Application Interface Specification". Open Mobile Alliance™. OMA-TS-PushCAI-V1_1. URL:http://www.openmobilealliance.org/

[PushCBS] “Push Over the Air Technical Specification – CBS Adaptation” . Open Mobile Alliance™. OMA-TS-Push_CBS_Adaptation-V1_0. URL:http://www.openmobilealliance.org/

[PushMO] "Push Management Object". Open Mobile Alliance™. OMA-TS-Push_MO-V1_1. URL: http://www.openmobilealliance.org/

[PushMsg] "Push Message Specification". Open Mobile Alliance™. OMA-TS-Push_Message-V2_3. URL: http://www.openmobilealliance.org/

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

[PPGServ] “Push Proxy Gateway Service”, Open Mobile Alliance™. OMA-TS-PPGService-V2_3. URL: http://www.openmobilealliance.org/

[RFC1918] “Address Allocation for Private Internets”, Y. Rekhter et al. 1996. URL: http://www.ietf.org/rfc/rfc1918.txt

[RFC1952] “GZIP file format specification version 4.3”, P. Deutsch. 1996. URL: http://www.ietf.org/rfc/rfc1952.txt

[RFC2104] “HMAC: Keyed-Hashing for Message Authentication”. H. Krawczyk et al. February 1997. URL: http://www.ietf.org/rfc/rfc2104.txt

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

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

[RFC2373] "IP Version 6 Addressing Architecture", R. Hinden et al. The Internet Society, 1998. URL: http://www.ietf.org/rfc/rfc2373.txt

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

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

[RFC3261] "SIP: Session Initiation Protocol". J. Rosenberg et al. The Internet Society, 2002. URL:http://www.ietf.org/rfc/rfc3261.txt

[RFC3264] "An Offer/Answer Model with the Session Description Protocol (SDP)". Rosenberg et al. URL:http://www.ietf.org/rfc/rfc3264.txt

[RFC3513] "Internet Protocol Version 6 (IPv6) Addressing Architecture", §§2.2, 2.3. R. Hinden et al. The Internet Society, 2003, URL:http://www.ietf.org/rfc/rfc3513.txt

[RFC3695] "Compact Forward Error Correction (FEC) Schemes", M. Luby, L. Vicisano. The Internet Society, 2004. URL: http://www.ietf.org/rfc/rfc3695.txt

[RFC3926] "FLUTE - File Delivery over Unidirectional Transport". T. Paila et al. October 2004. URL: http://www.ietf.org/rfc/rfc3926.txt

[RFC3986] "URI Generic Syntax" . T. Berners-Lee et al. The Internet Society, 2005. URL: http://www.ietf.org/rfc/rfc3986.txt

[RFC4028] "Session Timers in the Session Initiation Protocol (SIP)", S. Donovan, J. Rosenberg, April 2005. URL: http://www.ietf.org/rfc/rfc4028.txt

[RFC4483] E. Burger, Ed., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", RFC 4483, May 2006

[RFC4975] "The Message Session Relay Protocol (MSRP)", Campbell, B., Ed., Mahy, R., Ed., and C. Jennings, Ed., September 2007, URL:http://www.ietf.org/rfc/rfc4975.txt

[RFC5627] "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)". J. Rosenberg, October 2009, URL:http://www.ietf.org/rfc/rfc5627.txt

[RFC791] "Internet Protocol". DARPA, 1981, URL:http://www.ietf.org/rfc/rfc791.txt

[SEC_CF] "Enabler Release Definition for Application Layer Security Common Functions". Open Mobile Alliance™. OMA-ERELD-SEC_CF-V1_0. URL: http://www.openmobilealliance.org/

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

[SIPPush] "Push using SIP". Open Mobile Alliance™. OMA-TS-SIP_Push-V1_0. URL: http://www.openmobilealliance.org/

[TS23.003] "Technical Specification Group Core Network and Terminals; Numbering, addressing and identification". 3rd Generation Partnership Project (3GPP). URL:http://www.3gpp.org

[TS23.041] "Technical realization of Cell Broadcast Service (CBS)". 3rd Generation Partnership Project (3GPP). URL:http://www.3gpp.org

[TS26.346] "Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS)". 3rd Generation Partnership Project (3GPP). URL: http://www.3gpp.org

[TS33.246] "Security of Multimedia Broadcast/Multicast Service (MBMS)". 3rd Generation Partnership Project (3GPP). URL:http://www.3gpp.org

[UAPROF] "OMA User Agent Profile ", Open Mobile Alliance™, OMA-UAProf-v2_0. URL: http://www.openmobilealliance.org/

[UAProfSchema] OMA User Agent Profile Schema (latest), Open Mobile Alliance™,

http://www.openmobilealliance.org/Technical/schemas.aspx

[OMATLS] "OMA TLS Profile", Open Mobile Alliance™, OMA-TS-TLS-V1_0, URL:http://www.openmobilealliance.org/

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

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

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

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

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

2.2 Informative References

[ERELDDM] "Enabler Release Definition for Device Management version 1.2". Open Mobile Alliance. OMA-ERELD-DM-V1_2-20060208-C.

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

[PROVARCH] "Provisioning Architecture Overview 1.1". Open Mobile Alliance_.OMA-WAP-ProvArch-v1_1. URL: http://www.openmobilealliance.org/

[Push2.1] “Enabler Release Definition for Push Version 2.1”, Open Mobile Alliance™. OMA-ERELD-Push-V2_1. URL: http://www.openmobilealliance.org/

[Push2.2] “Enabler Release Definition for Push Version 2.2”, Open Mobile Alliance™. OMA-ERELD-Push-V2_2. URL: http://www.openmobilealliance.org/

[OMADICT] “Dictionary for OMA Specifications”, Open Mobile Alliance™, OMA-ORG-Dictionary, URL:http://www.openmobilealliance.org/

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

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	An implementation of a related set of functions that perform useful work, often enabling one or more services [OMADICT]
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	A device, user agent, or other entity that acts as the receiver of a service [OMADICT]
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	Digitized work that is processed, stored, or transmitted. It includes such things as text, presentation, audio, images,video, executable files, etc. Content may have properties such as media type, mime type, etc [OMADICT]
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	Equipment which is normally used by users for communications and related activities. The definition can be extended to cover remote monitoring applications where there is no user present, but the communications to and from the remote monitor use the same communications channels as when used by users [OMADICT]
End-user	see "user"
Point-to-Multipoint Push	Push content delivery to a group of users through the OTA-PTM Push-OTA protocol variant.
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 Client –Application Interface	A device-internal interface provided by Push Clients, via which Push applications can register for Push services with application-specified options, and receive notifications of Push events.
Push Channel	A Push content resource identified by a URI.
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	An entity or service that intiates Push content delivery to Push clients [OMADICT]
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 gateway acting as a Push proxy for Push Initiators, providing over-the-air Push message delivery

	services to Push clients [OMADICT]
Push Session	An active point-to-point transport protocol session over which push operations can be executed.
Push Whitelist	a list stored in the Terminal of PPG addresses and/or SMSC addresses that are authorised to push Content to the Terminal.
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	An entity that provides resources to clients in response to requests [OMADICT]
Terminal	see "client".
Terminal-ID	an identifier that is used by a PPG to uniquely identify a terminal.
User	An entity which uses services. Example: a person using a device as a portable telephone [OMADICT]
User agent	Any software or device that acts on behalf of a user, interacting with other entities and processing resources [OMADICT]
WAP Push	Push content delivery to a specific user via the WAP1 (OTA-WSP) or WAP2 (OTA-HTTP) Push-OTA protocol variants.
XML	The Extensible Markup Language is a World Wide Web Consortium (W3C) standard for Internet markup language, of which WML is one such language [OMADICT]

3.3 Abbreviations

ABNF	Augmented Backus-Naur Form
BCAST	OMA Broadcast Services
CPI	Capability and Preference Information
CBS	Cell Broadcast Service
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
MBMS	Multimedia Broadcast/Multicast Service
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-PTM	(Push) OTA over Point-to-Multipoint
OTA-SIP	(Push) OTA over SIP
OTA-WSP	(Push) OTA over WSP
PDP	Packet Data Protocol
PDU	Protocol Data Unit
PI	Push Initiator

PO-TCP	PPG Originated TCP connection establishment method
PPG	Push Proxy Gateway
PTM	Point-to-Multipoint
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
SIP	Session Initiation Protocol
SIR	Session Initiation Request
SMS	Short Message Service
SMSC	Short Message Service Centre
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

OMA Push Over-the-Air (Push-OTA) refers to a set of protocols and their bindings to various network bearers, via which a Push Proxy Gateway (PPG) or other Push Server interacts with a Push Client, to enable delivery of network-initiated content to Client Applications served by the Push Client. The OMA Push enabler architectural model and the enabler entities (PPG and Push Client), are introduced in [PushArch].

Figure 1 illustrates the OMA Push architectural model:

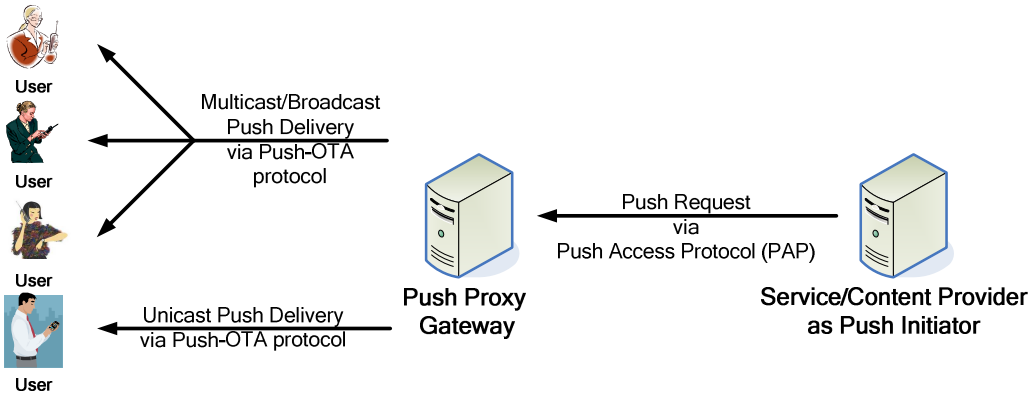


Figure 1: The Push Architectual Model

This specification focuses on the interface between the PPG and Push Client. Since there are a variety of protocol variants and bearer bindings supported by Push-OTA, a brief introduction to Push-OTA is provided here.

The Push-OTA protocol provides both “connectionless” and “connection-oriented” services. Connection-oriented service refers to a service context in which the Push Client has established a specific transport layer “connection” with the PPG, for reception of Push-based services. Connection-oriented service is supported via the WAP1 Wireless Session Protocol (WSP) as the OTA-WSP Push protocol, via the WAP2 HTTP-based OTA-HTTP Push protocol, and via the SIP-based OTA-SIP Push protocol. Connectionless service does not depend upon a pre-established/specific connection between the Push Client and PPG, and is supported via OTA-WSP, OTA-SIP, and OTA-PTM.

OTA-PTM is the protocol variant used in Point-to-Multipoint Push (PTM-Push), referring to Push operation across point-to-multipoint bearers, which in Push 2.3 includes MBMS, CBS, and BCAST. OTA-PTM is considered to support connectionless service. While the Push Client actively listens for Push events over the OTA-PTM bearers, there is no specific dialog created between the Push Client and PPG.

An important adjunct to connection-oriented service is the Session Initiation Application (SIA), via which connections can be intiated through Session Initiation Requests (SIR) delivered over connectionless Push services.

Figure 2 below provides a detailed end-to-end view of the service environment supported by the OMA Push enabler.

The following OMA Push-defined aspects of Figure 2 are described in further detail in this specification:

- Architectural entities: Push Client and Push Proxy Gateway
- Interfaces
  - Push Over-the-Air protocol variants
    - OTA-WSP: point-to-point delivery based upon WAP1 transport protocols
    - OTA-HTTP: point-to-point delivery based upon WAP2 transport protocols
    - OTA-SIP: point-to-point delivery based upon SIP transport protocols

- OTA-PTM: point-to-multipoint delivery based upon MBMS, OMA BCAST, and Cell Broadcast Service (CBS) as transport protocols

The following external entities provide functions and interfaces used by the Push Client and/or Push Server, for the adaptation of Push-OTA protocol variants:

- On the network side
  - Short Message Service Centers (SMSC) via which OTA-WSP/SMS is provided
  - Cell Broadcast Centers (CBC), via which OTA-PTM/CBS is provided
  - SIP/IP Core networks, via which OTA-SIP is provided
  - MBMS Broadcast Multicast Service Centers (BM-SC), via which OTA-PTM/MBMS is provided
  - OMA BCAST Service Distribution/Adaptation, via which OTA-PTM/BCAST is provided
- On the client side
  - SMS Clients, via which OTA-WSP/SMS abd OTA-PTM/CBS is delivered
  - OMA BCAST Clients, via which OTA-PTM/BCAST is delivered
  - MBMS Clients, via which OTA-PTM/MBMS is delivered

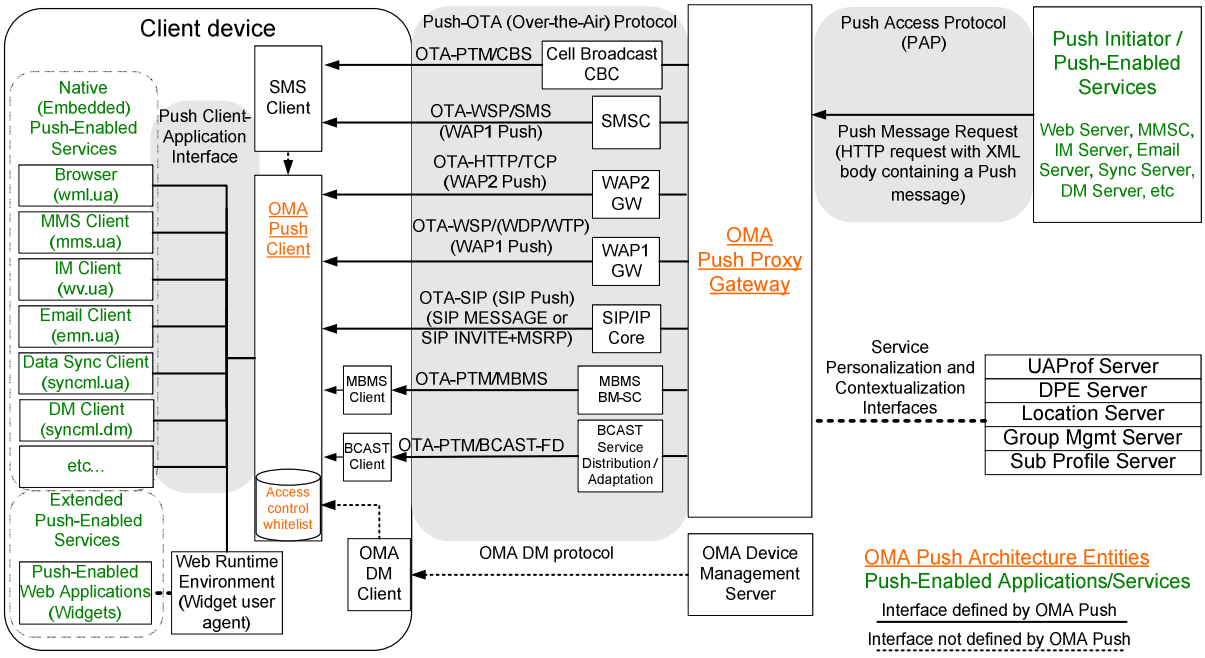


Figure 2 End-to-End Push Framework

4.1 Version 2.2 Functionality

This enabler release continues on the work of the OMA in the area of Push and is an extension of the [Push2.1] Enabler release, defining push security mechanisms and OTA-SIP as a new Push-OTA protocol variant. An aspect of the defined push security mechanisms depend on device management object extension defined in [PushMO] which depends on the OMA Device Management Enabler.

In addition this enabler release definition defines a minimum level of conformance for segmentation and re-assembly for SMS based Push, as well as push initiator guidelines on the most efficient way to use this form of push delivery mechanism.

4.2 Version 2.3 Functionality

This enabler release is an extension of the [Push2.2] Enabler release, and is referred to as Push 2.3 Point-to-Multipoint Push (PTM-Push). PTM-Push adds multipoint distribution methods to complement the existing point-to-point methods, enabling Push content delivery to a large number of clients simultaneously via network bearers supporting multicast and broadcast operation, e.g. MBMS, Cell Broadcast Service (CBS), and OMA BCAST.

PTM-Push defines push security mechanisms for use multipoint service contexts, and OTA-PTM as a new Push-OTA protocol variant. Aspects of the defined push security mechanisms and Push Client configuration depend on a device management object extension defined in [PushMO] which depends on the OMA Device Management Enabler [DMSTDOBJ] .

5. Protocol Variants

The Push OTA Protocol can be implemented to run on top of:

- WSP [[W-TCP]], as described in section 6
- HTTP 1.1 [RFC2616], as described in section 7
- SIP as described in section 8
- Point-to-Multipoint (PTM) bearer protocols MBMS, OMA BCAST, and CBS, as described in section 9

The WSP variant, referred to as "OTA-WSP", provides for both connection-orientated and connectionless push. Connection-orientated OTA-WSP operates over WTP/IP, and connectionless OTA-WSP operates over WDP/SMS or WDP/IP.

The HTTP variant, referred to as "OTA-HTTP", only provides functionality for connection-orientated push, and operates over TCP/IP. If TLS [OMATLS] 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".

The SIP variant, referred to as “OTA-SIP”, provides functionality for connection-orientated push and connectionless push. Security for OTA-SIP is provided by the underlying SIP/IP Core network as described in [SIPPush]. Connection-orientated OTA-SIP operates over the SIP/(UDP or TCP) control plane and MSRP/TCP/IP in the user plane, and connectionless OTA-SIP operates over the SIP/(UDP or TCP) control plane.

The PTM variant, referredto as “OTA-PTM”, provides connectioness push service. The three variants operate over BCAST/FLUTE/IP, MBMS/TCP/IP, and WSP/CBS. Security for content delivered via OTA-PTM is provided by the underlying bearer transport protocols.

A terminal MUST support one of the connection-orientated services provided by OTA-WSP, OTA-HTTP, or OTA-SIP when connection-orientated push is implemented, and MAY support more than one. A Push Proxy Gateway MAY implement all variants in order to support a wide range of mobile terminals. The PPG MAY support the connectionless services provided by OTA-WSP, OTA-SIP, or OTA-PTM. A terminal that supports a connectionless bearer (e.g. SMS) MUST support one of the the connectionless services provided by OTA-WSP, OTA-SIP, or OTA-PTM.

The following figure shows the protocol stack of the Push-OTA variants.

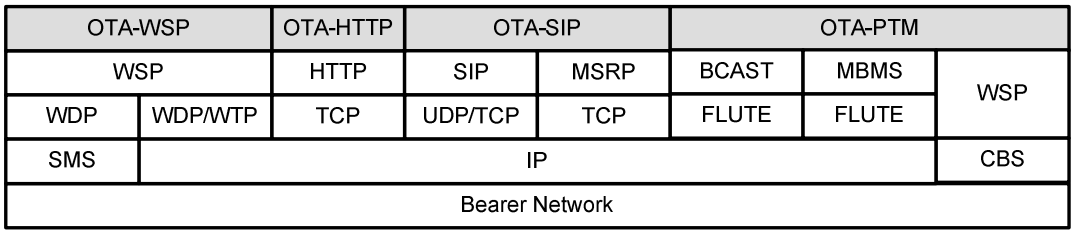


Figure 3 Pust-OTA Protocol Variant Stack

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

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

6.1 Service Primitive Definition

6.1.1 Notations

Notations for primitives and parameters follow the notations defined in [W-TCP].

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 4 demonstrates the layer-to-layer communication through the primitives.

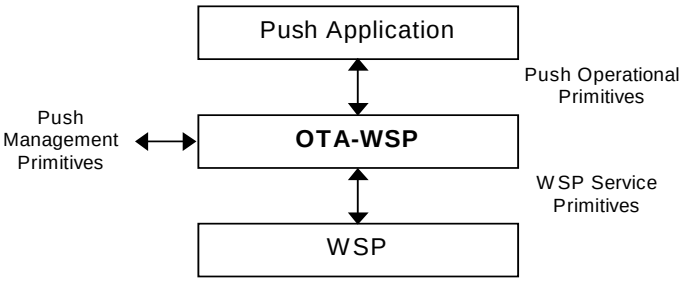


Figure 4: Illustration of Layer to Layer Communication

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-orientated 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.

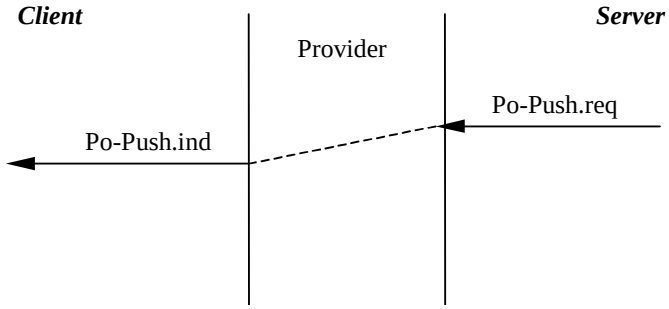


Figure 5: Unconfirmed Push

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-orientated 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 [W-TCP].

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

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 [W-TCP].

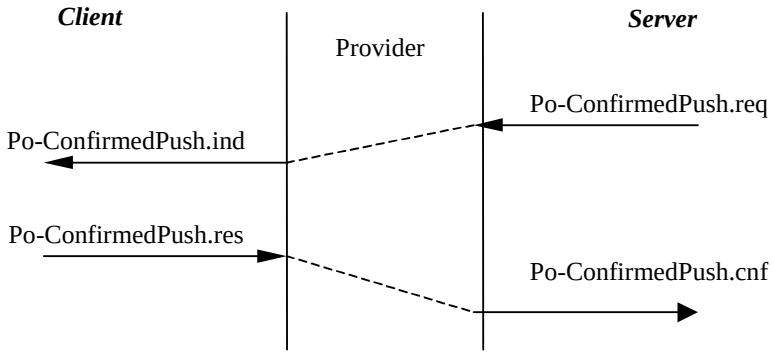


Figure 6: Confirmed Data Push

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 [W-TCP]. 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 [W-TCP].

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 [W-TCP] 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 [W-TCP].

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 [W-TCP].

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 [W-TCP].

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 [W-TCP].

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 [W-TCP].

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 [W-TCP]. 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-orientated* push as described in subsequent sections.

6.2.1 Connectionless Push

The connectionless push must be performed through WSP S-Unit-Push [W-TCP], 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.

6.2.1.1 Connectionless Push over SMS

In the case where connectionless push is to be performed over an SMS bearer type [WDP] then care needs to be taken over the size of payload to be pushed. If the payload to be pushed using this bearer is large then it will be automatically segmented over a number of smaller protocol PDUs [WDP].

In the case where there are a large number of these smaller PDU's operational difficulties (volume of messages, delays in transit etc.) as well as concatenation issues in the mobile device (buffering, timeouts etc.) may occur.

It is therefore recommended that the overall size of payload SHOULD be restricted to no more than 4 SMS messages. The most efficient use of the SMS bearer type is to restrict the message payload such that it may be encapsulated in a single SMS message. The server MUST support segmentation of *at least* 4 SMS messages.

A client MUST support the reassembly of *at least* 4 SMS messages.

The restriction on payload size does not apply to experimental content types or services.

The limits of the push content payload SHOULD be managed by a PPG and are dependent on the bearer type, service and push initiator.

6.2.2 Connection-orientated Push

The connection-orientated 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) [W-TCP] 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-orientated 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-orientated 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-orientated 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 [RFC2373] 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-orientated 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 device, the device is not referenced as "client" in the subsequent sections, but instead 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 [OMATLS]. A PPG implementing OTA-HTTP-TLS MUST implement the TLS server features specified by [OMATLS]. 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-orientated 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)

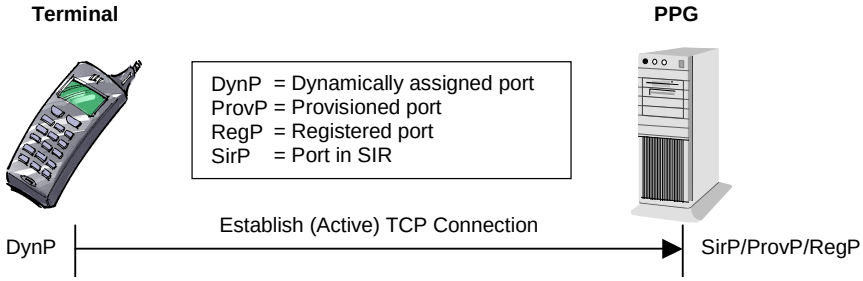


Figure 7: The TO-TCP method

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)

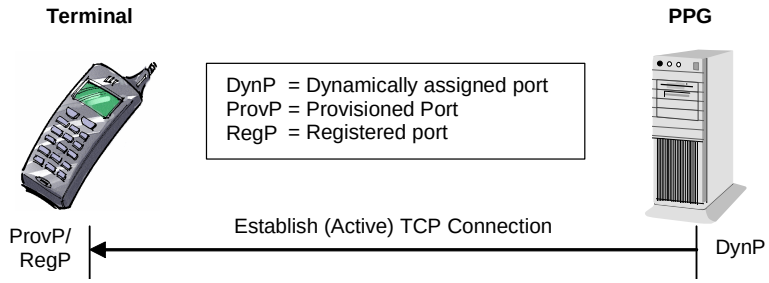


Figure 8: The PO-TCP method

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.

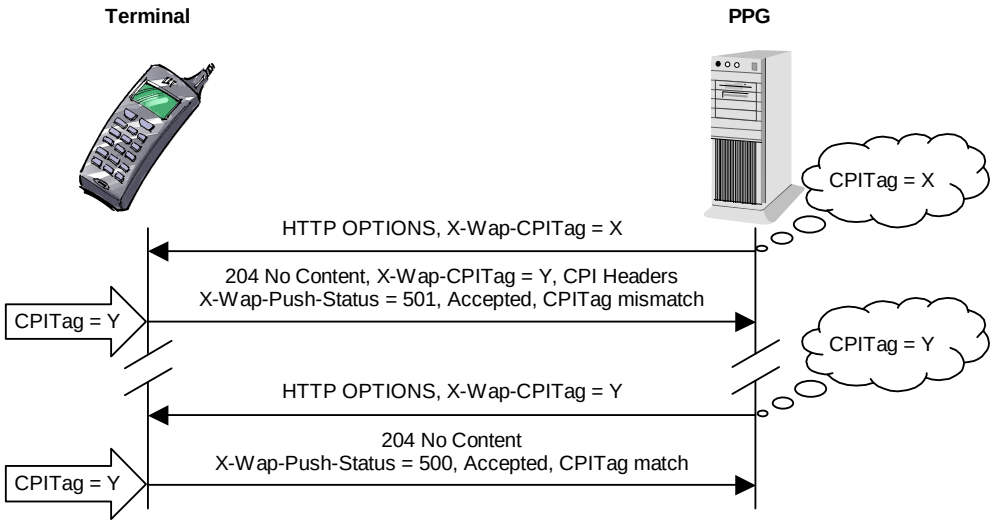


Figure 9: Registration (OPTIONS) request example

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.

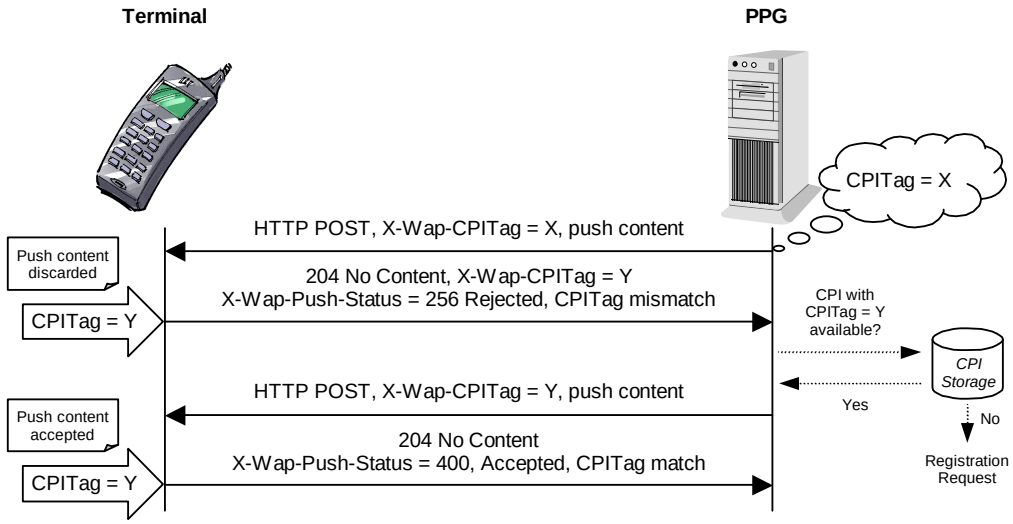


Figure 10: Registration validation (POST) example

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 [RFC2373] 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 [RFC2373] 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 [RFC2373].

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 [RFC2373] 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:

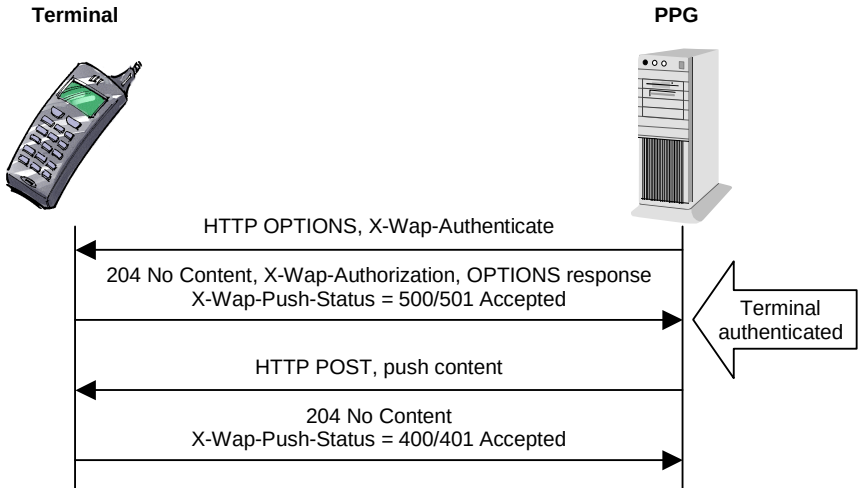


Figure 11: Terminal accepts unauthenticated registration request

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:

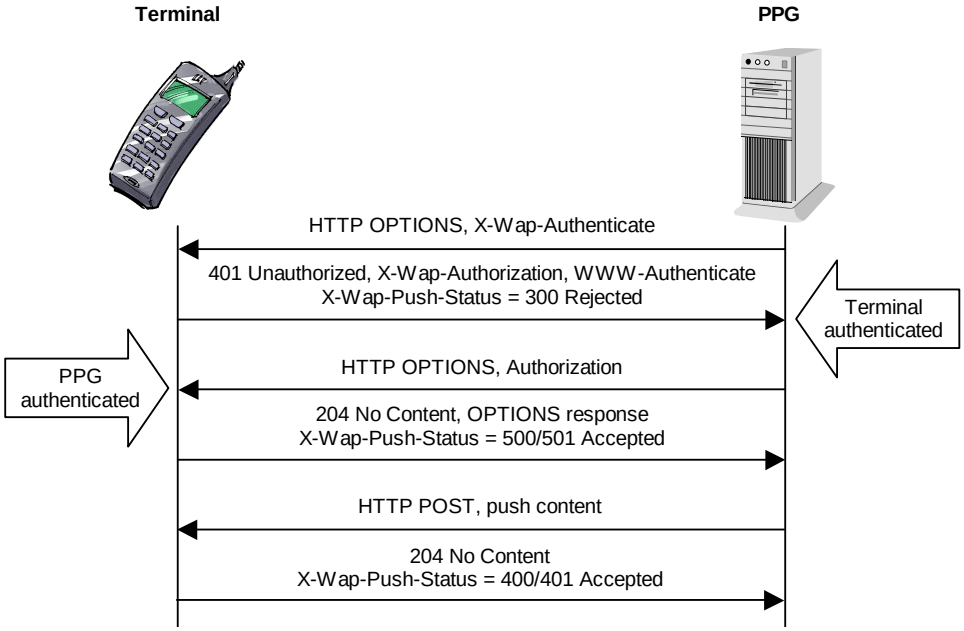


Figure 12: Terminal requests PPG authentication prior registration

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.

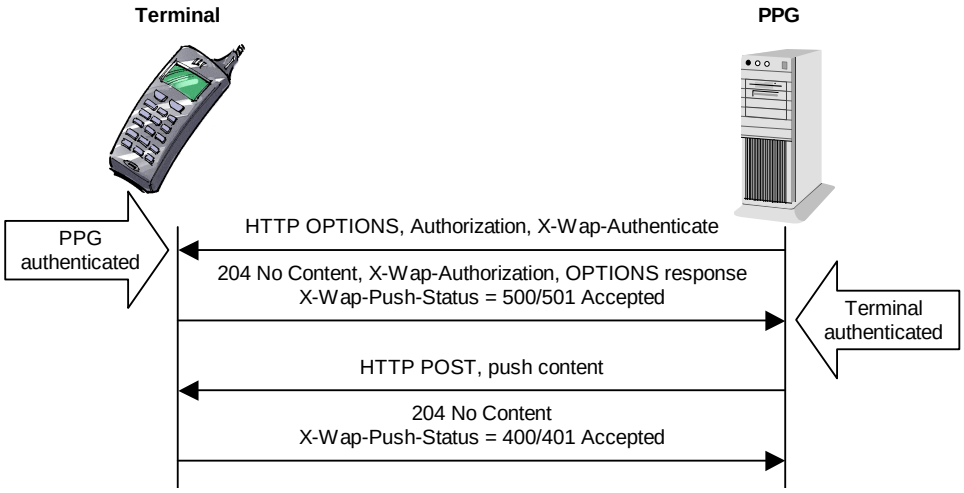


Figure 13: Use of "Basic" authentication

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 defined in PushMsg. It 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:

- `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.

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 [RFC2373] 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 [RFC2373] 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 10.1).

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 [RFC2373] 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. Push OTA Protocol over SIP (OTA-SIP)

See [SIPPush] for the general description of SIP-based Push protocols specified for use by OMA enablers. The following subsections modify and extend the requirements defined in [SIPPush], for the enabler-specific details of OTA-SIP as a SIP Push based enabler.

Except as modified and/or extended here:

- OTA-SIP supporting terminals (referred to subsequently as Push Clients) MUST implement the functions of Push Receiver Agent as described in [SIPPush].
- OTA-SIP supporting PPGs MUST implement the functions of Push Sender Agent as described in [SIPPush].

8.1 Protocol Overview

See [SIPPush] for an overview of SIP-based Push protocols specified for use by OMA enablers.

8.2 Protocol Description

8.2.1 IMS Communication Service Identifier

When the SIP/IP Core corresponding to 3GPP IMS or 3GPP2 MMD networks, the IMS Communication Resource Identifier (ICSI) “urn:urn-7:3gpp-service.ims.icsi.omapush” MUST be used as a tag value within the +g.3gpp.icsi-ref media feature tag by Push Clients and PPGs, in all circumstances in which the ICSI is specified to be included.

(editor note: urn:urn-7:3gpp-service.ims.icsi.omapush registration with 3GPP is pending)

8.2.2 Registration

The Push Client MUST register with the SIP/IP Core upon device startup, or when possible.

If a Push Client has access to an SMS-capable network, it MAY postpone registration until necessary or until directed to register by reception of a SIR. A Push Client which no longer has access to an SMS-capable network MUST register with the SIP/IP Core as soon as possible. If a Push Client has access to an SMS-capable network, it MAY postpone registration until necessary or until directed to register by reception of a SIR.

The REGISTER message MUST follow the rules and procedures as defined in SIP Push.

SIP REGISTER messages MUST include a Push Resource Identifier of each active push application (i.e. that is ready to receive Push messages) in the Contact header, as values of feature tag g.oma.pusheventapp.

8.2.3 SIP OPTIONS

8.2.3.1 Push Client

SIP OPTIONS messages MUST be sent immediately after successful registration, or after receiving a SIR request.

SIP OPTIONS messages MUST be sent to all of the PPG’s in the Push Whitelist, or the default PPG if the Push Whitelist is not defined.

SIP OPTIONS messages MUST follow the rules and procedures as defined in SIP Push.

SIP OPTIONS messages MUST include a Push Resource Identifier for the targeted push application in the Accept-Contact header, as the value of feature tag g.oma.pusheventapp.

The Push Client MAY include specific Push Resource Identifier in a subset of the PPG's to which SIP OPTIONS is sent, e.g. to relate push applications with specific PPG's. If only one SIP OPTIONS is sent, it MUST relate all registered push applications to the target PPG.

When the SIP/IP Core corresponds to 3GPP IMS or 3GPP2 MMD networks, the IMS Communication Resource Identifier (ICSI) "urn:urn-7:3gpp-service.ims.icsi.omapush" MUST be included by the Push Client in the OPTIONS message as a tag value within the +g.3gpp.icsi-ref media feature tag.

If compressions is supported the compression capabilities SHOULD be disclosed in the Accept-Encoding header as specified in [RFC3261].

8.2.3.2 PPG

The PPG MUST verify that the SIP OPTIONS message has the g.oma.pusheventapp feature tag defined in the Accept-Contact header.

The PPG MUST respond to a SIP OPTIONS message according to the rules and procedures as defined in SIP Push.

If the PPG is not able to support a specific Push Application ID requested in SIP OPTIONS, the PPG SHALL exclude the rejected Push Application ID from its response to the SIP OPTIONS. Reasons for rejection are implementation/service specific, e.g. detection of an application conflict between multiple registered terminals of the same user.

When receiving a SIP OPTIONS message the PPG MUST, unless a third-party REGISTER is delivered, subscribe to the reg-event package for the P-Asserted-Identity in the SIP OPTIONS message from the SIP/IP core defined in the Via header.

8.2.4 SIP MESSAGE Method (Pager-Mode Push)

The SIP MESSAGE method is used in OTA-SIP for connectionless push.

8.2.4.1 PPG

The SIP MESSAGE message handling MUST follow the rules and procedures as defined in SIP Push.

A PPG MAY know the registration status of a Push Client, e.g. via 3rd-party registration reg-event package subscription as defined in [SIPPush], or other unspecified means. A PPG that is unsure of the registration status of a Push Client SHOULD NOT send a SIP MESSAGE to a Push Client until a new SIP OPTIONS message has been received. A PPG that knows the registration status of a Push Client MAY send SIP MESSAGE without prior reception of SIP OPTIONS.

When the SIP/IP Core corresponds to 3GPP IMS or 3GPP2 MMD networks, the IMS Communication Resource Identifier (ICSI) "urn:urn-7:3gpp-service.ims.icsi.omapush" MUST be included by the PPG in the SIP MESSAGE as a tag value within the +g.3gpp.icsi-ref media feature tag.

SIP MESSAGE for Pager-Mode Push MUST include a Push Resource Identifier for the targeted push application in the Accept-Contact header, as the value of feature tag g.oma.pusheventapp.

SIP MESSAGE for Pager-Mode Push requests MUST contain Push Content embedded as described in section 8.4.1.

> When the PPG has positive knowledge that the push client supports compression (for example through SIP OPTIONS or) then the content MAY be compressed before sending. If compression is used then a compression method supported by the push client MUST be used and the Content-Encoding header as defined in [RFC3261] MUST be set accordingly.

The PPG MAY consider reception of 200 OK in response to SIP MESSAGE as a confirmation of push message delivery to the Push Client, but not to the targeted push application.

8.2.4.2 Push Client

The SIP MESSAGE message handling MUST follow the rules and procedures as defined in SIP Push.

The Push Client MUST deliver the content of the message to the application indicated by the value of the Push Resource Identifier and IMS Application Reference Identifier, if specified.

Note: If the Push client is not able to deliver the content to the application there will no information about this sent back to the PPG, as the 200 OK is sent back on on the acceptance of the SIP MESSAGE not on the delivery to the application.

8.2.5 SIP INVITE/MSRP Method (Session-Mode Push)

The SIP INVITE/MSRP method is used in OTA-SIP for connection-orientated push.

8.2.5.1 PPG

The SIP INVITE message handling MUST follow the rules and procedures as defined in SIP Push.

A PPG MAY know the registration status of a Push Client, e.g. via 3rd-party registration reg-event package subscription as defined in [SIPPush], or other unspecified means. A PPG that is unsure of the registration status of a Push Client SHOULD NOT send a SIP INVITE to a Push Client until a new SIP OPTIONS message has been received. A PPG that knows the registration status of a Push Client MAY send SIP INVITE without prior reception of SIP OPTIONS.

The SIP INVITE message SHALL include the option tag 'timer' in the Supported header according to rules and procedures of [RFC4028]

The SIP INVITE message SHOULD include the Session-Expires header with the refresher parameter set to "uac" according to rules and procedures of [RFC4028] . The “uac” value indicates that the PPG is responsible for refreshing the session if it exceeds the session timer. The PPG SHALL support session refresh as described in [RFC4028].

SIP INVITE requests sent to setup an MSRP session for Session-Mode Push MUST include a Push Resource Identifier for the targeted push application in the Accept-Contact header, as the value of feature tag g.oma.pusheventapp.

In the SIP INVITE, the PPG MUST include the following SDP parameters as defined in draft-ietf-mmusic-file-transfer-mech and as clarified in this section:

- Protocol Version (“v=”), Origin (“o=”), Session Name: “s=”, Connection Data (“c=”), Times: “t=0 0”, and Media Announcements (“m=”)
- Fixed Session Attributes (the same for every MSRP session)
    - “a=setup: passive”
    - “a=sendonly”
    - “a=accept-types: message/vnd.oma.push *”
- Variable Session attributes (specific to the MSRP session)
    - “a=path:” set to the MSRP session URI of the PPG
    - “a=file-transfer-id:” set to a new globally unique random identifier value associated with the current Push transaction
    - “a=file-selector” with selectors
        - “name:” set to the name of the push message to be sent by the PPG
        - “type:a” set to the MIME type of the push message to be sent via MSRP
        - “size:” set to the size of the push message
        - “hash:” set to the SHA-1 hash of the push message
    - “a=file-date:” with value for the push message

The “a=accept-types” attribute:

- MUST includethe “message/vnd.oma.push” MIME type
- MUST include the “message/cpim” MIME type or “*” per [RFC4975]
- MAY include other MIME types accepted by the PPG.

The file-selector “name” attribute MAY be included by the PPG, e.g. as a human-readable description for the Push transaction.

The file-selector “type” attribute MUST be set to the MIME type of the Push content.

The file-selector “size” attribute MUST be set to the size in bytes of the Push content.

The file-selector “hash” attribute MAY be included, set to the SHA-1 hash of the push message. If present, the Push Client MAY use the hash to validate the Push content later received in the MSRP session.

If a related PAP push request [PushPAP] contained a “Last-Modified” header, the “a=file-date” attribute MUST contain an equivalent “modification” parameter. Otherwise, the “a=file-date” attribute MAY contain an “creation” or “modification” value as appropriate for the push request.

MSRP messages sent by the PPG MUST contain a single push message embedded as described in section 8.4.1.

Push messages MUST be sent in MSRP SEND requests.

Push messages MAY be sent in several chunks as specified in [RFC4975].

When the SIP/IP Core corresponds to 3GPP IMS or 3GPP2 MMD networks, the IMS Communication Resource Identifier (ICSI) “urn:urn-7:3gpp-service.ims.icsi.omapush” MUST be included by the PPG in the SIP INVITE as a tag value within the +g.3gpp.icsi-ref media feature tag. For confirmed Push, the PPG SHALL request delivery confirmation by adding a Success-Report header set to yes in the MSRP SEND message.

The PPG MAY add a response token to the X-Wap-Push-Info header and a Success-Report header set to yes in the MSRP SEND message to enable Push Client responses with content body in MSRP REPORT. In this case the push message content MUST be encapsulated into a message/vnd.oma.push media type as specified in [PushMsg].

When the PPG has positive knowledge that the push client supports compression (for example through SIP OPTIONS or UAProf) then the content MAY be compressed by the PPG. If compression is to be used then the compressed push message content MUST be encapsulated into a message/vnd.oma.push media type as specified in [PushMsg]. The Content-Encoding header MUST be set according to the compression method used.

If the PPG receives MSRP REPORT including a “Status: 000 200 OK” line, the PPG SHALL consider this as indication of successful delivery to the targeted push application. Conversely, if the MSRP REPORT includes a non-200 status code, the PPG SHALL consider this as a failure to deliver the push message to the targeted push application.

The PPG SHALL ignore unexpected MSRP REPORT messages.

The PPG SHALL accept a content body in a MSRP REPORT message if an optional response was allowed per the X-Wap-Push-Info header.

8.2.5.2 Push Client

The SIP INVITE message handling MUST follow the rules and procedures as defined in SIP Push.

The Push Client MUST deliver the content of the MSRP message to the application indicated by the value of the Push Resource Identifier and IMS Application Reference Identifier (if specified) of the SIP INVITE.

If the Push Client accepts the SIP INVITE, it MUST send the following SDP parameters in the a SIP 200 OK response as defined in draft-ietf-mmusic-file-transfer-mech and as clarified in this section:

- Protocol Version (“v=”), Origin (“o=”), Session Name: “s=”, Connection Data (“c”), Times: “t=0 0”, and Media Announcements (“m=”)
- Fixed Session Attributes (the same for every MSRP session)
  - “a=setup: active”
  - “a=recvonly”
  - “a=accept-types: message/vnd.oma.push *”
- Variable Session attributes (specific to the MSRP session)
  - “a=path:” set to the MSRP session URI of the Push Client
  - “a=file-transfer-id:” as sent by the PPG
  - “a=file-selector” as sent by the PPG
  - “a=file-date:” as sent by the PPG

The “a=accept-types” attribute:

- MUST include the “message/vnd.oma.push” MIME type
- MUST include the “message/cpim” MIME type or “*” per [RFC4975]
- MUST include “*” or the specific MIME types accepted by the Push Client for the targeted push application

The file-transfer-id SDP attribute MAY be used by the Push Client to recognize a Push message it has already received. If the Push Client recognizes that it has already received a Push message, it MAY reject the current SDP offer by setting the port number to 0 in the “m=” line, as described in draft-ietf-mmusic-file-transfer-mech.

If the Success-Report header in the MSRP SEND request is set to yes, and the push message is both received completely and accepted by the target application, the Push Client MUST send an MSRP REPORT including a “Status: 000 200 OK” line. The MSRP REPORT MUST refer to the entire push message.

If a X-Wap-Push-Info header with a response token and a Success-Report header set to yes is present in the MSRP SEND message, the Push Client MAY add a content body to the MSRP REPORT.

If the Failure-Report header in the MSRP SEND request is set to yes, and the push message was either not received completely or not accepted by the application, the Push Client MUST send an MSRP REPORT including an appropriate non-200 message. The MSRP REPORT MUST refer to the entire push message.

8.3 Application Addressing

Application addressing for OTA-SIP occurs via the “Push Resource Identifier” described in [SIPPush].

8.4 Push Message Delivery

This section describes the methods of delivering push messages in OTA-SIP.

8.4.1 Embedded Push Message

If there is any push message header which can’t be mapped to an equivalent header in the SIP MESSAGE or MSRP SEND then the push message SHOULD be embedded into a “message/vnd.oma.push” media type as described in PushMsg. Otherwise the message MUST be directly embedded as the entity body of the SIP MESSAGE or MSRP SEND. The MIME type of the push message MUST be indicated by the Content-Type header of the SIP MESSAGE or MSRP SEND. Any push message headers MUST be mapped to equivalent headers of the SIP MESSAGE or MSRP SEND if direct embedding is done. Such push message headers may include those present in a related PAP request [PushPAP], defined by [PushMsg], or defined for use with a specific push content type. Push message headers that cannot be mapped to equivalent headers of the SIP MESSAGE or MSRP SEND include at least:

- `“Last-modified”:` for push messages sent via SIP MESSAGE. For push messages sent via MSRP, this header is mapped to the “file-date” SDP attribute in SIP INVITE.
- `“X-Wap-Initiator-URI”:` like all legacy WAP Push-OTA headers, this header is not compatible for use in SIP or MSRP directly.

8.4.1.1 The X-Wap-Push-Info Header

The X-Wap-Push-Info header is defined in PushMsg. It is a push message header sent by the PPG to provide the terminal with the following indications regarding each particular push transaction:

- `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. The terminal MUST NOT include any message body in the response if this token is not present.

Since Push messages sent via the SIP MESSAGE method do not support a message body in the response, the response token MUST NOT be sent if the SIP MESSAGE is used, and MUST be ignored by a OTA-SIP Push Client if received.

8.4.2 Indirect Push Message

Content indirection can be used with SIP MESSAGE, e.g. for content too large to be carried in the SIP MESSAGE (>1300 bytes). When using content indirection, the PPG MUST indirectly reference the Push message through use of the message/external-body Content-Type header in the SIP MESSAGE per [RFC4483], and include the Content-Type header for the Push message included in the SIP MESSAGE body, e.g.

```
Content-Type: message/external-body; access-type="URL";
    expiration="Mon, 24 June 2002 09:00:00 GMT";
    URL="http://psa.domain.com/a3cd20b8ae7f"
<CRLF>
Content-Type: text/vnd.wap.si
<CRLF>
```

The PPG MAY include other applicable headers for the Push message in the SIP MESSAGE body, as defined by [RFC4483]. If there are non mappable headers then the indirection SHOULD be embedded into a “message/vnd.oma.push” message, e.g.

```
Content-Type: message/vnd.oma.push
<CRLF>
X-Wap-Initiator-URI: http://www.yourfriendlyserviceprovider.com/yourmail
Content-Type: message/external-body; access-type="URL";
    expiration="Mon, 24 June 2002 09:00:00 GMT";
    URL="http://psa.domain.com/a3cd20b8ae7f"
<CRLF>
Content-Type: text/vnd.wap.si
<CRLF>
```

8.4.3 Examples

8.4.3.1 Push Request via SIP MESSAGE

Below is an example of a push request containing a Service Indication that allows the user to invoke email service via a browser.

```
MESSAGE sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" SIP/2.0
Via: SIP/2.0/UDP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
P-Asserted-Identity: sip:psa@domain.com
From: <sip:psa.domain.com>;tag=49583
```

```
To: “Bob” <sip:user@domain.com;tag=456248 ;gr= urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.omapush
Supported: gruu
Accept-Contact: *; +g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.omapush",
+g.oma.pusheventapp="wml.ua”
Call-ID: 723459815768908@998sdasdh09
CSeq: 1 MESSAGE
Date: Tue, 31 Jul 2001 10:13:05 GMT
Content-Language: en
Content-Type: text/vnd.wap.si
Content-Length: 301
<CRLF>
<?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>
<CRLF>
```

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

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP proxy.domain.com;branch=z9hG4bK123dsghds;received=192.0.2.1
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse;received=192.0.2.4
Max-Forwards: 70
From: <sip:psa.domain.com>;tag=49583
To: Bob <sip:user@domain.com> ;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
Supported: gruu
Call-ID: 723459815768908@998sdasdh09
CSeq: 1 MESSAGE
Content-Length: 0
```

8.4.3.2 Push Request via MSRP

Below is an example of a push request via MSRP, containing a Service Indication that provides some text and a link.

The PPG sends the SIP INVITE to the Push Client. The GRUU URI parameter (“gr=…”) is included if a GRUU was assigned during registration. The Call-ID is set per the Call-ID assigned by the Push Client when it registered. The requirements of the implementing enabler/application define the values of the Accept-Contact feature tags for the ICSI (`g.3gpp.icsi-ref`) and Push Resource Identifier (`g.oma.pusheventapp`). Note: the “s=” line of the SDP body is set as recommended in [RFC3264].

SIP INVITE as sent to the SIP/IP Core

```
INVITE sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" SIP/2.0
Via: SIP/2.0/TCP psa@domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
P-Asserted-Identity: <sip:psa@domain.com>
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>
From: <sip:psa.domain.com>;tag=49583
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.omapush
Supported: gruu
Contact: <sip:psa.domain.com>;+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-
service.ims.icsi.omapush"; +g.oma.pusheventapp="wml.ua”
```

```
Accept-Contact: *; +g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.omapush";
+g.oma.pusheventapp="wml.ua”
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 INVITE
Content-Type: application/sdp
Content-Length: 467
<CRLF>
v=0
o=psa 2890844526 2890844526 IN IP4 domain.com
s=-
c=IN IP4 psa.domain.com
t=0 0
m=message 7654 TCP/MSRP *
a=setup: passive
a=sendonly
a=accept-types: application/vnd.oma.push *
a=path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
a=file-transfer-id: a3cd20b8ae7f
a=file-selector: name:"pushid-a3cd20b8ae7f" type:text/vnd.wap.si
 size:4092 hash:sha-1:72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=file-date:modification:"Tue, 31 Jul 2001 10:13:00 GMT"
```

The SIP/IP Core forwards the SIP INVITE to the Push Client:

SIP INVITE as forwarded to the Push Client

```
INVITE sip:user@192.0.2.4 SIP/2.0
Via: SIP/2.0/TCP proxy.domain.com;branch= z9hG4bK123dsghds
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 69
P-Asserted-Identity: <sip:psa@domain.com>
P-Called-Party-ID: sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6"
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>
From: <sip:psa.domain.com>;tag=49583
Supported: gruu
Contact: <sip:psa.domain.com>;+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-
service.ims.icsi.omapush"; +g.oma.pusheventapp="wml.ua”
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 INVITE
Date: Sun, 21 May 2006 13:02:03 GMT
Content-Type: application/sdp
Content-Length: 467
<CRLF>
v=0
o=psa 2890844526 2890844526 IN IP4 domain.com
s=-
c=IN IP4 psa.domain.com
t=0 0
m=message 7654 TCP/MSRP *
a=setup: passive
a=sendonly
a=accept-types: application/vnd.oma.push *
a=path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
a=file-transfer-id: a3cd20b8ae7f
a=file-selector: name:"pushid-a3cd20b8ae7f" type:text/vnd.wap.si
 size:4092 hash:sha-1:72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=file-date:modification:"Tue, 31 Jul 2001 10:13:00 GMT"
```

The Push Client analyses the SDP parameters and returns the agreed parameters by sending a SIP 200 "OK" response to the SIP/IP Core. If the Push Client for some reason does not want to accept the Push content, it sets the port number of the media (“m=”) line associated with the file to zero.

SIP 200 OK as sent to the SIP/IP Core

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP proxy.domain.com;branch=z9hG4bK123dsghds;received=192.0.2.1
Via: SIP/2.0/TCP psa@domain.com;branch=z9hG4bK776sgdkse;received=192.0.2.4
Max-Forwards: 70
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Supported: gruu
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 INVITE
Contact: <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>;
+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.omapush";
+g.oma.pusheventapp="wml.ua”
Content-Type: application/sdp
Content-Length: 483
<CRLF>
v=0
o=bob 2890844656 2890844656 IN IP4 domain.com
s=-
c=IN IP4 userpc.domain.com
t=0 0
m=message 8888 TCP/MSRP *
a=setup: active
a=recvonly
a=accept-types: application/vnd.oma.push text/vnd.wap.si application/vnd.oma.sic *
a=path: msrp://userpc.domain.com:8888/9di4ea;tcp
a=file-transfer-id: a3cd20b8ae7f
a=file-selector: name:"pushid-a3cd20b8ae7f" type:text/vnd.wap.si
 size:4092 hash:sha-1:72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=file-date:modification:"Tue, 31 Jul 2001 10:13:00 GMT"
```

The SIP/IP Core forwards the SIP 200 "OK" response to the PPG.

SIP 200 OK as forwarded to the PPG

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP psa@domain.com;branch=z9hG4bK776sgdkse
P-Asserted-Identity: user@domain.com
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
Supported: gruu
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 INVITE
Contact: <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>;
+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.omapush";
+g.oma.pusheventapp="wml.ua”
Content-Type: application/sdp
Content-Length: 483
<CRLF>
v=0
o=bob 2890844656 2890844656 IN IP4 domain.com
s=-
c=IN IP4 userpc.domain.com
t=0 0
m=message 8888 TCP/MSRP *
a=setup: active
a=recvonly
a=accept-types: application/vnd.oma.push text/vnd.wap.si application/vnd.oma.sic *
a=path: msrp://userpc.domain.com:8888/9di4ea;tcp
```

```
a=file-transfer-id: a3cd20b8ae7f
a=file-selector: name:"pushid-a3cd20b8ae7f" type:text/vnd.wap.si
 size:4092 hash:sha-1:72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=file-date:modification:"Tue, 31 Jul 2001 10:13:00 GMT"
```

The PPG acknowledges the SIP 200 “OK” response with a SIP ACK request sent to the SIP/IP Core.

SIP ACK as sent to the SIP/IP Core

```
ACK sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 SIP/2.0
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 ACK
Content-Length: 0
```

The SIP/IP Core forwards the SIP ACK request to the PPG.

SIP ACK as forwarded to the Push Client

```
ACK sip:user@192.0.2.4  SIP/2.0
Via: SIP/2.0/TCP proxy.domain.com;branch= z9hG4bK123dsghds
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 69
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 198456 ACK
Content-Length: 0
```

In this example, the Push content is large enough to require chunking over multiple MSRP SEND requests, and a delivery success report is requested. The PPG opens a TCP connection to the Push Client, then the PPG sends the the first chunk of data in a MSRP SEND request to the Push Client using the MSRP channel, including and the Success Report header is inserted and set to yes.

MSRP SEND as sent to the Push Client

```
MSRP d93kswow SEND
To-Path: msrp://userpc.domain.com:8888/9di4ea;tcp
From-Path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-1038/1641
Success-Report: yes
Content-Type: text/vnd.wap.si
<CRLF>
<?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.jabberwocky.com/carroll/walrus.html"
    created="2001-07-31T10:13:00Z"
    si-expires="2001-08-07T10:13:00Z">
"The time has come," the Walrus said,
"To talk of many things:
Of shoes--and ships--and sealing-wax--
Of cabbages--and kings--
And why the sea is boiling hot--
And whether pigs have wings."
```

```
"But wait a bit," the Oysters cried,
"Before we have our chat;
For some of us are out of breath,
And all of us are fat!"
"No hurry!" said the Carpenter.
They thanked him much for that.

"A loaf of bread," the Walrus said,
"Is what we chiefly need:
Pepper and vinegar besides
Are very good indeed--
Now if you're ready, Oysters dear,
We can begin to feed."

"It was so kind of you to come!
And you are very nice!"
The Carpenter said nothing but
"Cut us another slice:
I wish you were not quite so deaf--
I've had to ask you twice!"
-------d93kswow+
```

The Push Client responds with an MSRP 200 “OK” to the first MSRP SEND request to the PPG using the MSRP channel. Note that MSRP supports pipelining of SEND requests, i.e. the sender does not have to wait for a 200 OK to send the next chunk. This example shows the case in which pipelining is not used.

MSRP 200 OK as sent to the PPG

```
MSRP d93kswow 200 OK
To-Path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
From-Path: msrp://userpc.domain.com:8888/9di4ea;tcp
-------d93kswow$
```

The PPG sends the the second next chunk of data in a MSRP SEND request to the Push Client using the MSRP session.

MSRP SEND as sent to the Push Client

```
MSRP op2nc9a SEND
To-Path: msrp://userpc.domain.com:8888/9di4ea;tcp
From-Path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
Message-ID: 12339sdqwer
Byte-Range: 1039-1641/1641
Success-Report: yes
Content-Type: text/vnd.wap.si

"It seems a shame," the Walrus said,
"To play them such a trick,
After we've brought them out so far,
And made them trot so quick!"
The Carpenter said nothing but
"The butter's spread too thick!"

"I weep for you," the Walrus said:
"I deeply sympathize."
With sobs and tears he sorted out
Those of the largest size,
Holding his pocket-handkerchief
Before his streaming eyes.

"O Oysters," said the Carpenter,
"You've had a pleasant run!
Shall we be trotting home again?'
```

```
But answer came there none--
And this was scarcely odd, because
They'd eaten every one.
</indication>
</si>
------- op2nc9a$
```

The Push Client responds with an MSRP 200 “OK” on the second MSRP SEND request to the PPG using the MSRP session.

MSRP 200 OK as sent to the PPG

```
MSRP op2nc9a 200 OK
To-Path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
From-Path: msrp://userpc.domain.com:8888/9di4ea;tcp
------- op2nc9a$
```

As requested by the PPG, the Push Client sends a MSRP REPORT conveying the status of the received message.

MSRP REPORT as sent to the PPG

```
MSRP dkei38sd REPORT
To-Path: msrp://psa.domain.com:7654/a3cd20b8ae7f;tcp
From-Path: msrp://userpc.domain.com:8888/9di4ea;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-1908/1908
Status: 000 200 OK
------- dkei38sd$
```

Because the PPG and the Push Client no longer need the MSRP session, the PPG sends a SIP BYE to SIP/IP Core to disconnect the MSRP session with the Push Client.

SIP BYE as sent to the SIP/IP Core

```
BYE sip:user@domain.com gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" SIP/2.0
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583>
Call-ID: 390458126334871@998sdasdh09
CSeq: 198457 BYE
Content-Length: 0
```

The SIP/IP Core forwards the SIP BYE request to the Push Client.

SIP BYE as forwarded to the Push Client

```
BYE sip:user@192.0.2.4  SIP/2.0
Via: SIP/2.0/TCP proxy.domain.com;branch= z9hG4bK123dsghds
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 69
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 198457 BYE
Content-Length: 0
```

The PPG responds to the Push Client with SIP 200 "OK".

SIP 200 OK as sent to the SIP/IP Core

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP proxy.domain.com;branch=z9hG4bK123dsghds;received=192.0.2.1
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 2 BYE
Content-Length: 0
```

The SIP/IP Core forwards the SIP 200 "OK" to the PPG.

SIP 200 OK as forwarded to the PPG

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP psa.domain.com;branch=z9hG4bK776sgdkse
To: Bob <sip:user@domain.com;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6>;tag=456248
From: <sip:psa.domain.com>;tag=49583
Call-ID: 390458126334871@998sdasdh09
CSeq: 2 BYE
Content-Length: 0
```

9. Point-to-Multipoint Push OTA Protocol (OTA-PTM)

9.1 Common Aspects

This section described aspects of OTA-PTM that are common to the various transport bearer bindings described in the following sections.

9.1.1 Client Application Registration with Push Client

The Push Client MAY enable Client Applications to register for OTA-PTM based services over the Push-CAI interface as described in [PushCAI]. During the registration process, the Push Client MAY allow Client Applications to request a specific OTA-PTM protocol binding, and supply related access parameters for delivery of Push content.

The Push Client SHALL reject registration of a Client Application for a specific OTA-PTM protocol binding, and provide a distinct error response under the following conditions:

- the terminal is not does not support the specific OTA-PTM protocol binding ("not supported")
- the terminal is not configured for service via the specific OTA-PTM protocol binding ("not configured")
- either the Client Application is not authorized to activate OTA-PTM based services, or the specified access parameters are not allowed ("not allowed")

Upon successful registration of a Client Application, the Push Client SHALL initiate content reception as applicable for the activated OTA-PTM protocol bindings.

The Push Client SHALL allow Client Applications that have registered for specific OTA-PTM protocol bindings via the Push-CAI interface to terminate their registration, either by updating their registration to remove a specific OTA-PTM protocol binding, or by deregistering completely. Upon successful deregistration of a Client Application, the Push Client SHOULD terminate any Push content reception arrangements for the affected OTA-PTM protocol bindings, which were associated solely with the deregistered Client Application. The intent is that deregistration of a specific Client Application does not affect Push service delivery to other Client Applications.

9.1.2 Push Channel

The term Push Channel is used here to refer to the configuration of a communication path via which a PPG delivers Push messages to a Push Client, which overall represents a Push content resource as delivered by the PPG to Push Clients. The Push Channel is defined by the PPG Service Provider, and can be associated with various key attributes that are common to all Push content delivered via the Push Channel:

- A registration URI: a resource from which the Push Client can obtain configuration information as necessary for access to the Push Channel
- Push-Application-Id: OMNA-registered or unregistered (application-defined) identifiers of the Push-enabled applications to which the Push content should be delivered, once it is received by the terminal
- Transport bearer-specific configuration parameters, for each transport bearer to which the Push Channel is bound, e.g. multicast, broadcast, and unicast bearers
- Target Push Client group: any arbitrary association of users or terminals to the Push service represented by the Push Channel

The Push Channel is intended to enable OMA Push entities (PPG and Push Client) to bind to the underlying transport bearers for specific Push-enabled services. To avoid cross-layer dependencies, the underlying transport bearers and associated network/device entities are expected to be unaware of the Push Channel, or the use of the transport bearer for OMA Push. In particular, protocol data of the underlying transport bearers will have no semantic relationship to the OMA Push enabler. The OMA Push entities are responsible for specific bindings to the transport bearers, and are solely aware of the association of those bindings with specific Push-enabled services/applications.

Note that the Push Channel defined by OTA-PTM is different from the "FLUTE channel" defined by [TS26.346], which per [TS26.346] is "defined by the combination of a sender and an address associated with the channel by the sender". The Push Channel represents a set of configuration options for the OMA Push enabler, as compared to the FLUTE channel which is a transport bearer configuration.

The PPG MAY associate a Push Channel to any combination of transport bearers. A single transport bearer configuration SHALL be used for each associated transport bearer, For example, a Push/MBMS Push Channel should be associated with a single FLUTE channel, and a Push/CBS channel associated with a single CBS Message-Id.

The PPG MAY associate a target Push Client group to each Push Channel.

The PPG MAYassociate a registration URI to each Push Channel. The URI SHALL be globally unique, and include:

- the URI scheme name "http:"
- a URI hierarchial part starting with "//" and including
    - a URI authority part that is unique to the PPG, e.g. a registered Internet hostname
    - a URI path part that is unique to the Push Channel

The Push Channel URI MAY identify a Push-Application-Id, to associate content delivered in the Push Channel to a specific Push-enabled application, via:

- a URI query part that includes the parameter "push-app-id", which have the value of an OMNA-registered or unregistered (application-defined) Push Application Id, in the form of a URN or assigned number.

As an example, fully formed Push Channel URIs may look like

```
http://ppg1.domain.com/foo97531?push-app-id=urn:x-wap-application:wap.ua
```

```
http://ppg1.domain.com/foo97531?push-app-id=0x00
```

9.1.3 Push Service Registration

Push Service Registration is an optional procedure via which a Push Client obtains the necessary configuration parameters to complete access to a Push Service provided by a PPG. Push Service Registration can be provided by the PPG as a mechanism for Push Initiators to establish trust with Push Clients, by associating Push Service Registration addresses with services offered by specific Push Initiators.

PPG's and Push Clients MAY support Push service registration. This section applies to PPG's and Push Clients that support Push Service Registration.

PPG's SHALL define a Push Service Registration URI associated to each Push Service supported by the Push Service Registration procedure. Whether a specific Push Service requires Push Service Registration is a deployment decision. In some cases no explicit registration may be required, e.g. if the Push Client is preconfigured with all necessary options or obtains them through registration of a Client Application. The PPG MAY initiate the Push Service Registration procedure by sending a Session Initiation Request (SIR) as described in section 10.3. Other than by these methods, how the Push Client discovers Push Service Registration URI's is unspecified.

To initiate Push Service Registration, the Push Client SHALL send an HTTP POST to the Push Service Registration URI, including:

- the User-Agent request-header set to identify the host device (e.g. "vendor-model/version"), and the name and version of the Push Client as user agent initiating the request
- the Accept request-header, including at least the values "application/vnd.oma.push, multipart/related"
- the X-Wap-Profile request-header set to the URI of the User Agent Profile for the host device

- the Accept-Encoding request-header with value per the supported HTTP compression encodings, e.g. deflate and / or gzip
- the Content-Type entity-header with value “application/vnd.oma.push”
- a message-body formatted per the the following XML schema and
    - using the RegistrationRequest element
    - including the PushApplicationId, ApplicationId, and Accept attributes as applicable to the Client Application

PPG acceptance of a particular Push Service Registration is a service deployment issue, e.g. depending upon user authentication/authorization requirements defined by the service.

If the Push Service Registration request is not accepted, the PPG SHALL provide an appropriate HTTP error response.

If the Push Service Registration request is accepted, the PPG SHALL provide a 200 OK response, including:

- the Content-Type entity-header set according to the content in the response
    - If the response includes MBMS User Service Description information, the Content-Type SHALL be set to “multipart/related” and the type attribute set to “application/vnd.oma.push”.
    - If the response does not include MBMS User Service Description information, the Content-Type SHALL be set to “application/vnd.oma.push”.
- the RegistrationResponse element in the application/vnd.oma.push entity-body, including
    - A ConnectionProfile element for each applicable connection profile for the service, and including detailed elements/attributes as applicable for the specific service
        - A Protocol attribute as applicable to the connection profile: one of “OTA-WSP”, OTA-HTTP”, “OTA-SIP”, “OTA-PTM”, or other implementation-specific value
        - A Bearer attribute as applicable to the connection profile: one of the Push-SupportedBearers defined in [UAProfSchema], one of the new OTA-PTM related values “MBMS”, “BCAST”, or “CBS”, or other implementation-specific value
        - A Port attribute as applicable to the protocol and bearer. If the default port is to be used, this attribute may be absent.
        - A ContactPoint attribute as applicable to the protocol, e.g. for OTA-WSP, OTA-HTTP, or OTA-SIP, the IP address, hostname, or URI of the PPG
        - For OTA-PTM with CBS bearer, the CellBroadcastMessageId attribute [TS23.041]
        - For OTA-PTM with MBMS bearer, the MbmsUserServiceDescription attribute, with the value of the Content-ID of the multipart entity containing the User Service Description [TS26.346]
        - For OTA-PTM with BCAST bearer, the Bcast-Access-Info element [BCAST-SvcGuide]

The set of connection profiles that is provided in response is a deployment decision. For example, the PPG can be configured to respond only with a Push/CBS connection profile, or both a Push/CBS and Push/MBMS connection profile.

Upon successful Push Client registration to a Push Service, further actions by the PPG are implementation/deployment specific.

Upon reception of successful Push Service Registration response, further actions by the Push Client are implementation/deployment specific, e.g. interaction with the Client Application as described in [PushCAI].

To deregister from a Push Service, the Push Client SHALL send an HTTP POST to the Push Service Registration URI, including:

- the User-Agent request-header set to identify the host device (e.g. “vendor-model/version”), and the name and version of the Push Client as user agent initiating the request
- the Accept request-header, including at least the values “application/vnd.oma.push, multipart/related”
- the X-Wap-Profile request-header set to the URI of the User Agent Profile for the host device
- the Accept-Encoding request-header with value per the supported HTTP compression encodings, i.e. deflate and / or gzip
- the Content-Type entity-header with value “application/vnd.oma.push”
- a message-body formatted per the the following XML schema and
    - using the RegistrationRequest element
    - including the PushApplicationId and ApplicationId attributes as applicable to the Client Application

Upon Push Client deregistration from a Push Service, the PPG SHALL provide a 200 OK response. Further actions by the PPG are implementation/deployment specific.

Upon deregistration from a Push Service, further actions by the Push Client are implementation/deployment specific, e.g. interaction with the Client Application as described in [PushCAI].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.openmobilealliance.com/oma-push-registration/1.0"
    xmlns:push="http://www.openmobilealliance.com/oma-push-registration/1.0"
    xmlns:dcd="http://www.openmobilealliance.com/oma-dcd/1.0"
    elementFormDefault="qualified">
    <xsd:import namespace="http://www.openmobilealliance.com/oma-dcd/1.0"
    schemaLocation="http://www.openmobilealliance.org/Tech/profiles/dcd_xsd-v1_0.xsd"/>
    <xsd:element name="RegistrationOperationRequest">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="RegistrationRequest" type="push:RegistrationRequestType"/>
                <xsd:element name="DeregistrationRequest" type="push:RegistrationRequestType"/>
                <xsd:element name="RegistrationResponse" type="push:RegistrationResponseType"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="RegistrationRequestType">
        <xsd:attribute name="PushApplicationId" type="xsd:string" use="optional"/>
        <xsd:attribute name="ApplicationId" type="xsd:string" use="optional"/>
        <xsd:attribute name="Accept" type="xsd:string" use="optional"/>
        <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
    <xsd:complexType name="DeregistrationRequestType">
        <xsd:attribute name="PushApplicationId" type="xsd:string" use="optional"/>
        <xsd:attribute name="ApplicationId" type="xsd:string" use="optional"/>
        <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
    <xsd:complexType name="RegistrationResponseType">
      <xsd:sequence>
        <xsd:element name="ConnectionProfile" type="push:ConnectionProfileType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
    <xsd:complexType name="ConnectionProfileType">
        <xsd:sequence>
          <xsd:element name="Bcast-Access-Info" type="dcd:Bcast-Access-Info-Type" minOccurs="0"
maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="Protocol" type="xsd:string" use="optional"/>
        <xsd:attribute name="Bearer" type="xsd:string" use="optional"/>
        <xsd:attribute name="Port" type="xsd:string" use="optional"/>
```

```
        <xsd:attribute name="ContactPoint" type="xsd:string" use="optional"/>
        <xsd:attribute name="CellBroadcastMessageId" type="xsd:string" use="optional"/>
        <xsd:attribute name="MbmsUserServiceDescription" type="xsd:string" use="optional"/>
        <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
</xsd:schema>
```

9.1.4 Push Response Spreading

In PTM-Push operations received over broadcast bearers, for which the Push operation would normally result in immediate automated content retrieval from the network, Push Clients SHALL support Push response spreading to reduce the dynamic impact on networks and servers. To spread such responses, Push Clients SHALL delay the automatic response by a pseudo-randomized delay, recommended to be from zero to ten seconds. To reduce correlation of the random number across devices, the initial seed for the random number algorithm SHALL use device-unique information.

9.2 OTA-PTM over MBMS

The PPG and Push Client MAY support OTA-PTM over MBMS (“Push/MBMS”). This section applies to the PPG and Push Client that supports Push/MBMS.

PPG’s MAY incorporate BM-SC functions as defined [TS26.346]. Push Clients MAY incorporate MBMS receiver functions as defined [TS26.346]. For clarity, PPG’s and Push Clients which incorporate MBMS functions are referred to here as “MBMS-capable”. PPG’s and Push Clients which do not incorporate such functions, but interface with MBMS entities are referred to as “MBMS-aware”.

9.2.1 Protocol Overview

The PTM-Push Enabler provides a set of functions which enable the delivery of content over broadcast/multicast networks, including but not limited to: content delivery, file distribution, service protection, and content protection.

This section describes the necessary data and interactions needed to extend the Push Enabler to facilitate delivery over MBMS.

MBMS-aware Push Clients MUST be capable of interfacing with an MBMS receiver function in the device. As [TS26.346] does not specify an interface between the MBMS receiver function and applications in the device, the requirements for such an interface (implementation-specific) are described here in functional terms only.

The MBMS-capable Push Client MUST support FLUTE as described in [TS26.346] for Push/MBMS.

The PPG MAY incorporate BM-SC functions. The PPG which incorporates BM-SC functions MUST support FLUTE as described in [TS26.346] for PTM-Push over MBMS. [TS26.346] does not define an interface for the BM-SC toward the content source (in this case the PPG), thus this specification does not define a specific binding of the PPG to the BM-SC.

9.2.2 Content Encoding

Push content MAY be encoded by PPG’s using the generic GZIP algorithm [RFC1952]. Note for MBMS-aware PPG’s, whether the PPG or the BM-SC encodes content is a deployment decision.

MBMS-capable Push Clients SHALL support GZIP content decoding of Push content. Note for MBMS-aware Push Clients, content decoding is expected to be provided by the MBMS receiver function per [TS26.346].

For GZIP-encoded files, the “Content-Encoding” attribute of the file description in the FDT SHALL be given the value “gzip”.

9.2.3 Forward Error Correction Building Block

The "Compact No-Code” FEC scheme [RFC3695] (FEC Encoding ID 0, also known as "Null-FEC") SHALL be supported by MBMS-capable PPG's and Push Clients.

MBMS-capable Push Clients SHALL support the Raptor FEC scheme as defined in [TS26.346]. MBMS-capable PPG's MAY support the Raptor FEC scheme.

9.2.4 Push Channel setup

The Push enabler is only responsible for the delivery to the Push client. For MBMS content delivery the PPG SHALL define a Push Channel as described in section 9.1.2.

To set up a PTM-Push session using MBMS, MBMS-capable PPG's MUST use a MBMS User Service Description (USD) element as described in section 9.2.5. The PPG MAY directly announce the available channels as a "User Service Announcement over point-to-point push bearers" [TS26.346], using OTA-WSP, OTA-HTTP, or OTA-SIP for delivery of the service announcement. Note that while OTA-SIP is not referenced as a protocol binding by [TS26.346] for use with the "x-3gpp:mbms.service.announcement.ua" Push Application Id ([TS26.346] having preceded OMA SIP Push), Push Clients that support OTA-SIP can deliver the service announcement to the "MBMS user service announcement application" in the terminal.

It is assumed that MBMS-aware PPG's will rely upon a BM-SC to provide the service announcement.

A SDP description element MUST contain the relevant FLUTE channel the client should listen to.

Push Channel setup on the terminal side MAY occur transparently to the Push Client (e.g. between an MBMS-enabled application and the MBMS receiver function). Otherwise, e.g. if Push Channel setup for Push/MBMS is triggered by Client Application registration per [PushCAI], the Push Client MUST be capable of initiating the necessary procedures to activate the MBMS service, either directly or via the MBMS receiver function in the terminal. Such procedures MAY include Push Service Registration as described in section 9.1.3.

9.2.4.1 PPG

Prior to delivering content over Push/MBMS, the MBMS-capable PPG SHALL verify that a file delivery session is associated with the Push Channel for which the content is to be delivered. If a file delivery session is not associated with the Push Channel, the MBMS-capable PPG SHALL follow the procedures described in Section 9.2.4 to establish a Push/MBMS file delivery session for the channel.

The PPG SHALL support the message delivery result notification if requested by the Push Initiator, by at least one of the following methods:

- For MBMS-capable PPG's, via the "Reception Reporting" procedure as described in [TS26.346]
- For MBMS-aware PPG's, over the interface between the PPG and BM-SC

When delivering Push Content over MBMS, the PPG SHOULD verify that content for exactly one Push Channel is delivered over exactly one MBMS file delivery session. Any other approach for mapping Push Channels to MBMS sessions is considered out of scope for this specification.

MBMS-capable PPG's SHALL set the 'Content-Location' data element of the FLUTE FDT to the URI of the Push content file being delivered, according to the rules and procedures of [TS26.346].

MBMS-capable PPG's MAY use the Push Channel URI as the root of the "Content-Location" data element of the FLUTE FDT, to associate the specific file being delivered with the Push Channel and target Push-enabled application.

If the PPG adds any WAP-specific headers to the Push message, the Push message MUST be encapsulated into a message/vnd.oma.push media type as specified in [PushMsg]. The PPG MAY use unspecified additional procedures to determine when to encapsulate Push messages in the message/vnd.oma.push media type, e.g. based upon the need to pass message headers.

MBMS-capable PPG's SHALL be capable of sending Push Content over FLUTE to the Push Client.

9.2.4.2 Push Client

MBMS-capable Push Clients SHALL be capable of receiving Push Content over FLUTE.

9.2.5 User Service Description Element

MBMS-capable PPG's SHALL provide a reference to the 'sdp-description' element describing the PTM-Push access information associated with a Push channel in a User Service Description Element, as described in [TS26.346].

The following is an example of the SDP parameters used to set up a FLUTE session:

```
v=0
o=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24
s=File d    elivery session example
i=More information
t=2873397496 2873404696
a=mbms-mode:broadcast 1234 1
a=FEC-declaration:0 encoding-id=1
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=flute-tsi:3
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF1E:03AD::7F2E:172A:1E24/1
b=64
a=lang:EN
a=FEC:0
```

9.3 Push over BCAST

The PPG and Push Client MAY support OTA-PTM over BCAST ("Push/BCAST"). This section applies to the PPG and Push Client that supports Push/BCAST.

Push Clients MAY directly support the BCAST-related functions described in the following subsections, or interface with a BCAST client for these functions, through unspecified mechanisms.

Push content is delivered via Push/BCAST independently of the WAP Push headers, which are mapped to FLUTE FDT elements if possible. If mapping to FLUTE FDT elements is not possible, the Push content and WAP Push headers are enveloped in a MIME type application/vnd.oma.push.

9.3.1 BCAST File Delivery Session

As a prerequisite to Push/BCAST use, a BCAST File Delivery Session must have been established, to which both the PPG and targeted Push Clients have access. This section defines the overall requirements for PPG and Push Client establishment and termination of BCAST File Delivery Sessions for OTA-PTM.

Establishment or termination of BCAST delivery for Push content is assumed to be initiated the PPG service provider based upon unspecified service-specific conditions. The detailed mechanisms for BCAST File Delivery Session establishment are assumed to be completed via PPG service provider provisioning, and are not defined here.

A separate BCAST File Delivery Session SHOULD be provisioned for each Push Channel as described in section 9.1.2.

Push Clients SHOULD be capable of Push content reception via multiple distinct BCAST File Delivery Sessions, e.g. for service to different Client Applications or as a member of multiple target Push Client groups.

Successful establishment of a BCAST File Delivery Session will include identification of the necessary BCAST access information for Push Client configuration, described in section 9.3.4. This information can be delivered to the necessary Push Clients via OMA Device Management (DM) or Session Initiation Request (SIR).

If OMA DM is available on the device, the Push Client SHALL support configuration of BCAST access parameters as part of the OMA Push Management Object (MO) via OMA DM [ERELDDM] .

Upon successful configuration of BCAST access information for Push/BCAST, if at least one Push/BCAST related Client Application is ready to receive BCAST-delivered Push content, the Push Client SHALL initiate Push/BCAST content reception as described in section 9.3.2.1.

Push Client configuration to deactivate BCAST File Delivery Sessions for delivery of Push content, and removal of the BCAST access information, may occur via:

- Update of the OMA Push MO using OMA DM
- Expiration of the BCAST File Delivery Session as defined in the BCAST Service Guide
- Application-specific conditions, e.g. termination of a Push-enabled service or removal of the registered Client Application

9.3.2 Push Client

9.3.2.1 Initiating Push/BCAST Content Reception

If the BCAST File Delivery Session for which content reception is to be initiated is within its scheduled distribution window or does not have a schedule, the Push Client SHALL immediately initiate content reception for the file delivery session based upon the BCAST access information.

If the BCAST File Delivery Session for which content reception is to be initiated is not within its scheduled distribution window, the Push Client SHALL initiate content reception at the appropriate time based on the schedule associated with the BCAST access information.

9.3.2.2 Receiving Push/BCAST Content

When the Push Client receives a notification that a new FLUTE FDT instance as defined in BCAST10-Distribution is available for a Push/BCAST related File Delivery Session, the Push Client SHALL examine the FDT instance to determine whether the Push content should be retrieved from the file delivery session according to the procedures described below.

If a PPG Whitelist is configured and contains at least one Push/BCAST-related PPG source address (i.e. authority part of the Push Channel URI):

- If the new FLUTE FDT instance describes unretrieved or updated file(s) for which the ‘Content-Location’ attribute matches an allowed Push/BCAST PPG source address, the Push Client SHALL retrieve and process the file(s) as Push content.
- If the new FLUTE FDT instance describes unretrieved or updated file(s) for which the ‘Content-Location’ attribute does not match an allowed Push/BCAST PPG source address, the Push Client SHALL ignore the file(s).

If a PPG Whitelist is not configured or contains no Push/BCAST-related PPG source addresses, and the new FLUTE FDT instance describes unretrieved or updated file(s) for which the ‘Content-Location’ attribute matches a registered Push Channel, the Push Client SHALL retrieve and process the file(s) as Push content.

The Push Client SHALL use the procedures described in Section 5.2.4 of BCAST10-Distribution when examining the FLUTE FDT instance to determine whether an updated version of a file is available.

9.3.3 PPG

9.3.3.1 Delivering Push/BCAST Content

Prior to delivering Push content over BCAST, the PPG SHALL verify that a BCAST File Delivery Session is associated with the Push Channel for which the content is to be delivered.

If the PPG adds any WAP-specific headers to the Push message, the Push message MUST be encapsulated into a message/vnd.oma.push media type as specified in [PushMsg]. The PPG MAY use unspecified additional procedures to determine when to encapsulate Push messages in the message/vnd.oma.push media type, e.g. based upon the need to pass message headers.

The PPG SHALL add the Push content to the BCAST File Delivery Session associated with the targeted Push Client group as described in section 9.3.4.

9.3.4 FLUTE Transport Binding

The Push Client SHALL be capable of either directly receiving content via BCAST File Delivery Sessions, or interfacing with a BCAST File Delivery Client via unspecified mechanisms, for the purpose of receiving Push content over BCAST.

The PPG SHALL be capable of acting as a BCAST Service Application and interfacing with a BCAST BSD/A via unspecified mechanisms, for the purpose of delivering Push content over BCAST.

The usage of FLUTE is defined in [RFC3926] and BCAST10-Distribution and the FLUTE Schema for OMA BCAST is defined in [BCAST10-XMLSchema-FD-FDT].

In order to publish Push content to a BCAST File Delivery Session, the PPG SHALL create a FLUTE FDT File element for the document containing the Push content according to the rules and procedures of [RFC3926] and BCAST10-Distribution.

The PPG SHALL set the FLUTE ‘Content-Location’ attribute of the File element to the URI of the Push content file being delivered. The PPG SHALL MAY use the Push Channel URI as the root of the “Content-Location” data element of the FLUTE FDT, to associate the specific file being delivered with the Push Channel and target Push-enabled application.

The PPG SHALL set the FLUTE ‘Content-Type’ attribute of the File element to the appropriate MIME type for the Push content.

9.4 Push over CBS

The PPG and Push Client MAY support OTA-PTM over CBS (“Push/CBS”). This section applies to the PPG and Push Client that supports Push/CBS.

Push/CBS is described in [PushCBS].

10.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), that no SIP registration or event notification subscription exists (OTA-SIP), 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 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-orientated push.

The SIA content type contains separate lists of contact points for OTA-WSP and other protocols (OTA-HTTP and OTA-SIP). This implies that the originator of an SIR (usually a PPG) can choose to indicate any of them. The 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 other protocols (OTA-HTTP and OTA-SIP), 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, OTA-SIP, and in OTA-WSP respectively.

10.1 SIR in OTA-HTTP

10.1.1 Session Initiation Application

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

10.1.2 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 [ProvCont].

10.1.3 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).

10.2 SIR in OTA-SIP

10.2.1 Session Initiation Application

SIA MUST be supported both by terminals and PPGs implementing OTA-SIP. In order to ensure support for SIA in terminals that do not support SMS, the SIA can be invoked via OTA-WSP, OTA-HTTP, or OTA-SIP. Note that SIA delivery may not be possible if the terminal is not connected to the network via at least one supported bearer.

10.2.2 PPG Procedure

A PPG can instruct a terminal to establish a registration with a SIP IP/Core network and publish its capabilities by sending an SIR to the SIA in the terminal, indicating contact points for OTA-SIP. The SIA is addressed by its registered Application-ID [ProvCont].

The conditions that may trigger delivery of SIR in OTA-SIP are implementation and deployment specific, but may include for example PPG reception of a push request to a Push Client for which the PPG has no current information, e.g. the PPG has received no SIP OPTIONS from the Push Client. The PPG may thus determine that it needs to invoke the SIA on the terminal, in order to receive the Push Client's capabilities, and if needed, to be added to the Push Whitelist.

To deliver a SIR via OTA-SIP, the PPG SHALL embed the SIR in the body of a SIP MESSAGE request, following the rules and procedures as defined in [SIPPush]. In order to direct the SIP MESSAGE to a specific terminal (as one of a set of terminals registered for the user), the PPG SHALL include the GRUU for the specific terminal.

SIP MESSAGE for a SIR MUST include the Push Resource Identifier "push.sia" as the value of feature tag g.oma.pusheventapp in the Accept-Contact header, to indicate the targeted push application.

To deliver SIR via OTA-WSP, the PPG SHALL follow the procedure described in section 10.3.2.

To deliver SIR via OTA-HTTP, the PPG SHALL follow the procedure described in section 10.1.2.

To indicate that the terminal should contact the PPG via OTA-SIP, the PPG SHALL include the address of the PPG in the SIA Contact Points field entry with ProtOpts field set to the value for OTA-SIP.

10.2.3 Terminal Procedure

A terminal SHALL recognize an SIR for OTA-SIP by inclusion of a ***Contact Points*** field entry with ***ProtOpts*** field set to the value for OTA-SIP.

When acting upon an SIR for OTA-SIP, the terminal MUST take the following actions:

- Establish connectivity with the network, if not already done. For example if the terminal is not connected to an IP bearer network, it should establish the connection.
- Establish a registration with a SIP IP/Core network , if not already done. For example the Push Client may not be registered with the SIP/IP Core network, and must register before it can send SIP OPTIONS.
- Send an SIP OPTIONS request following the rules and procedures defined in SIP Push
    - To all of the PPG's in the Push Whitelist, or the default PPG if the Push Whitelist is not defined
    - If the Contact Point contains a URI, to the specified PPG by the URI

If multiple Contact Points (one or more PPGs) are included in the SIR, the terminal SHOULD send a SIP OPTIONS request to each PPG.

10.3 SIR in OTA-WSP

10.3.1 Session Initiation Application

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

10.3.2 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 [ProvCont].

10.3.3 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).

10.4 SIR in OTA-PTM

10.4.1 Session Initiation Application

SIA MUST be supported both by terminals and PPGs implementing OTA-PTM. In order to ensure support for SIA in terminals that do not support SMS, the SIA can be invoked via OTA-WSP, OTA-HTTP, or OTA-SIP. Note that SIA delivery may not be possible if the terminal is not connected to the network via at least one supported bearer.

10.4.2 PPG Procedure

A PPG can instruct a terminal to initiate the Push Service Registration procedure (see section 9.1.3) by sending an SIR to the SIA in the terminal, indicating the contact point for OTA-PTM. The SIA is addressed by its registered Application-ID [ProvCont].

The conditions that may trigger delivery of SIR in OTA-PTM are implementation and deployment specific, but may include for example PPG reception of a push request to a Push Client for which the PPG has no current information, e.g. the Push Client has not registered for an OTA-PTM service. The PPG may thus determine that it needs to invoke the SIA on the terminal, in order to initiate the Push Service Registration procedure.

To deliver SIR via OTA-SIP, the PPG SHALL follow the procedure described in section 10.2.2.

To deliver SIR via OTA-WSP, the PPG SHALL follow the procedure described in section 10.3.2.

To deliver SIR via OTA-HTTP, the PPG SHALL follow the procedure described in section 10.1.2.

To indicate that the terminal should contact the PPG via OTA-PTM, the PPG SHALL include the Push Service Registration URI in the SIA Contact Points field entry with ProtOpts field set to the value for OTA-PTM.

10.4.3 Terminal Procedure

A terminal SHALL recognize an SIR for OTA-SIP by inclusion of a ***Contact Points*** field entry with ***ProtOpts*** field set to the value for OTA-PTM.

When acting upon an SIR for OTA-PTM, the terminal MUST take the following actions:

- Establish connectivity with the network, if not already done. For example if the terminal is not connected to an IP bearer network, it should establish the connection.
- Initiate the Push Service Registration procedure as defined in section 9.1.3.

If multiple Contact Points (one or more Push Service Registration URIs) are included in the SIR, the terminal SHOULD initiate the Push Service Registration procedure to each registration address.

11.Security Considerations

11.1 Terminal Based Push Whitelists

In order to protect against denial of service attacks and push from unauthorized sources a Push Whitelist mechanism is defined. The Push Whitelist consists of a list of trusted Push sources, and the related Push applications/content that should be accepted from them. If the Push Whitelist is provisioned on the terminal, then the terminal MUST apply it to each Push message received. If the Push message is trusted according to the Push Whitelist, the Push Client MUST accept the push PDU and process it according to its content. If the Push Whitelist is not provisioned then the Push Client MUST accept the push PDU and process it according to its content.

The Push Whitelist can be provisioned into the terminal with the use of OMA Provisioning [PROVARCH] or OMA Device Management [DM-TND-V1-2]. In the case of devices which conform to [PROVARCH] the Push Whitelist MUST be managed using the APPLICATION characteristic as defined in the OMA provisioning specifications [ProvCont]. Note that if [PROVARCH] is used, only source addresses for PPG's and SMSC's can be configured in the Push Whitelist. In the case where the device conforms to [ERELDDM] the white list MUST be managed as a specific extension to the DM Tree [DM-TND-V1-2] as a management object [DMSTDOBJ] and is defined in the push management object specification [PushMO].

In the case where `APPLICATION` characteristic is used to provisioning the Push Whitelist, the APPID parameter MUST be equal to urn:oma:mo:oma-push:1.0. The ADDR parameter is used to carry SIP URI's, otherwise the `TO-PROXY` parameter MUST be used. The `TO-NAPID` parameter MUST be used to convey the E.164 address of one or many the trusted SMSCs.

Upon reception of an `APPLICATION` characteristic with the APPID parameter set to urn:oma:mo:oma-push:1.0, the Push Whitelist MUST be initiated with a new entry for every ADDR, `TO-PROXY` parameter, or `TO-NAPID` parameter [ProvCont]. For each entry the elements MUST be initiated in the following way:

1. The portion of the of the Push Whitelist containing trusted PPGs MUST be initiated with a new entry for each ADDR, PXLOGICAL.PXPHYSICAL.PXADDR, or PXLOGICAL.PXPHYSICAL.PXADDR-FQDN parameter that is listed in the APPLICATION characteristic.
2. The portion of the of the Push Whitelist containing trusted SMSCs MUST be initiated with a new entry for each NAPDEF.NAP-ADDRESS parameter that is listed in the APPLICATION characteristic.

If the PXADDR-FQDN parameter is available, the domain name of the PPG MUST be resolved to an IP-address before the address is stored in the Push Whitelist of trusted PPGs. If the domain name is updated with the use of Device Management, the Terminal MUST resolve the domain name to an IP address at least every time the domain name is updated. The Terminal MAY store the domain name and resolve the domain name with a higher frequency, e.g. every time a Push PDU is received or according to the DNS resolution response.

All other parameters in the `APPLICATION` characteristic not explicitly mentioned above MUST be discarded by the Terminal. The Terminal MUST be capable of handling at least 10 entries in the Push Whitelist.

Examples of the APPLICATION characteristics are shown below:

```
<characteristic type="APPLICATION">
  <parm name="APPID" value="oma:mo:oma-push:1.0"/>
  <parm name="TO-PROXY" value="ppg.foo.com"/>
  <parm name="TO-NAPID" value="NAPSMSC"/>
</characteristic>

<characteristic type="NAPDEF">
  <parm name="NAPID" value="NAPSMSC"/>
  <parm name="BEARER" value="GSM-SMS"/>
  <parm name="NAME" value="Operator foo SMSC"/>
  <parm name="NAP-ADDRESS" value="+35809503401"/>
  <parm name="NAP-ADDRTYPE" value="E164"/>
</characteristic>
```

```
<characteristic type="PXLOGICAL">
  <parm name="PROXY-ID" value="ppg.foo.com "/>
  <parm name="NAME" value="Operator foo PPG"/>
  <characteristic type="PXPHYSICAL">
    <parm name="PHYSICAL-PROXY-ID" value="PROXY 1"/>
    <parm name="PXADDR" value="215.221.51.5"/>
    <parm name="PXADDRTYPE" value="IPV4"/>
    <parm name="TO-NAPID" value="INTERNET"/>
  </characteristic>
</characteristic>

<characteristic type="APPLICATION">
  <parm name="APPID" value="oma:mo:oma-push:1.0"/>
  <parm name="ADDR" value="sip:ppg@atlanta.com"/>
</characteristic>
```

If the Push message is encapsulated in the message/vnd.oma.push media type as specified in [PushMsg], the Push message MUST be unencapsulated prior to validation to ensure the actual Content Type is checked as necessary.

On receipt of a Push PDU, the terminal MUST validate the PDU against the Push Whitelist, using the procedure below:

- If there is an entry in the Push Whitelist, matching the source address for OTA-WSP, OTA-HTTP, OTA-SIP, and OTA-PTM (for MBMS and BCAST bearers), or the Message Id for OTA-PTM (CBS)
    - If no Push Application Id values are provisioned, the Push message is validated
    - Otherwise if one of the provisioned Push Application Id's match the Push Application Id in the Push message, or the Push Application Id of the Client Application as registered
        - If no Content Type values are provisioned, the Push message is validated
        - Otherwise, if one of the provisioned Content Type values match the Content Type of the Push message, the Push message is validated
        - Otherwise the Push message is not validated
    - Otherwise the Push message is not validated
- Otherwise the Push message is not validated

The terminal MUST support prefix matching and '*' as a wildcard when comparing the source address, CBS Message Id, Push Application Id, and Content Type values. For example if the address type is E164 with the address formed into a string of the form <number-plan><type-of-number><digit>+, the address matches the corresponding half of the Push Whitelist entry if all the characters in the entry match all the leading characters of the address. In the event of this being a WAP 2 Push where the Push Proxy gives guidance to the device as to which provisioning context to use, via the X-Wap-Push-ProvURL then that context MUST be used. In OTA-SIP the P-Asserted-Identity header must be used to compare to the URIs in the Whitelist.

If Push message was not validated, the terminal MUST reject the Push PDU.

If there is no Push Whitelist configured in the terminal, the terminal MAY compare the origination address data to the push enabled physical proxies or access points provisioned on the device (PXPHYSICAL or NAPDEF[ProvCont]). In the event of this being a WAP 2 Push where the Push Proxy gives guidance to the device as to which provisioning context to use, via the X-Wap-Push-ProvURL then that context MUST be used.

If originating address does not match any address configured in the Push Whitelist or other connectivity configuration the terminal MUST reject the push PDU.

11.2 Processing Session Initiation Requests

In the case of SIR, 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 or OTA-SIP), 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.

11.3 Authenticating content sources in a connectionless environment

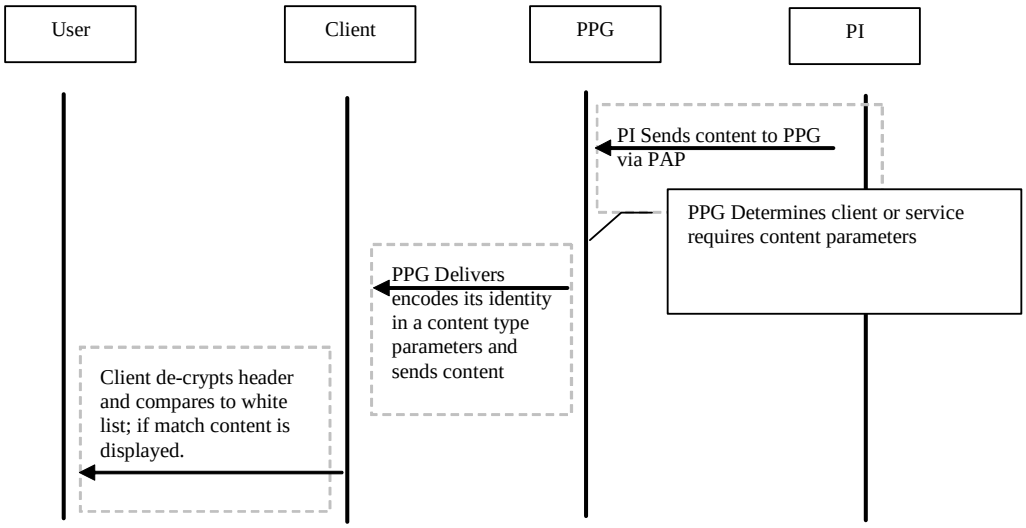
This section has already dealt with authenticating the source of the push message from a network perspective (either a PPG or intermediate SME / SMSC. A secondary mechanism MAY be used to verify the originating source of the content (either PI or PPG). In order to establish & authenticate a trust relationship between PPG and client the mechanism detailed in the Provisioning Bootstrap [PROVBOOT] is reused. In Bootstrap the content is trusted due to it being ‘signed’ using a shared secret. This shared secret may be user defined or it might be some specific information that is related to the bearer or network. The shared secret is then used to generate parameters to the provision content type *application/vnd.wap.connectivity-wbxml*; namely the parameters SEC and MAC.

SEC indicates the security mechanism that was used (user defined, network specific etc) and the MAC parameter indicates the authentication code calculated using the pushed data and the shared secret.

In the case where the shared secret is known to the PPG (or PI) the PPG (or PI) may generate the SEC and MAC parameters for the content type.

11.3.1 SEC & MAC Content Type Parameters

The use of SEC and MAC is as per OMA Provisioning Bootstrap specification [PROVBOOT]. The following example clarifies the usage in the case of Push OTA



The SEC and MAC parameters MUST have the same values as defined in section 5.2.1 of the Bootstrap specification [PROVBOOT].

SEC value

The SEC parameter can have the following values

Value	Meaning
1	USERPIN
2	USERNETWPIN
3	USERPINMAC

MAC value

Defined as follows: -

MAC = HMAC-SHA (K, D)

Where K is the Shared Secret and D is the data, in this case the content that is to be pushed to the targeted terminal and HMAC-SHA is the Keyed-Hashing for Message Authentication algorithm [RFC2104], which utilizes the SHA-1 hash function (supported by the device as defined in section 7.2.6).

The option of using NETWPIN, as defined in [PROVBOOT] *is not considered secure for the purposes of push.*

11.3.1.1 Terminal Processing

Once the SEC & MAC parameters have been calculated they should be added as parameters to the content type header value.

The content is then sent to the targeted terminal unencrypted with the new parameters added to the content type.

In the case of WSP push then these new parameters MUST be encoded as per **Appendix A** of the [W-TCP] spec and per OMA Naming Authority [ProvCont].

An example of a push message from PPG to device using the content type service indication, which is supported natively by the device, in a WAP 2 HTTP Push example, with added SEC and MAC parameters is shown below: -

```
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; sec=2; mac=8C8AEA662AED56A9F345909997147BC9886CD458
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>
```

In this example a USERNETWPIN is used as the key in the HMAC-SHA calculation. This specification does not provide a mechanism to establish the shared secret, it is assumed that this will be handled in implementation.

This example is *illustrative* only – it does not supersede the authorization mechanisms specified in section 7.2.6. If explicit authorization is available it MUST supersede the mechanism outlined here.

If the terminal does not understand the added content-type parameters, it MUST ignore them. However, if the terminal is capable of processing the parameters to the content type and they do not match the shared SEC parameter value the push MUST be rejected.

11.4 Security in Point-to-Multipoint Push

The user-specific security mechanisms described above (e.g. SEC and MAC) are not usable across point-to-multipoint bearers. Thus security in PTM-Push depends upon additional security mechanisms when multiple clients are targeted in

broadcast and multicast environments. These mechanisms are not defined by PTM-Push, but are either provided by the specific bearer binding for OTA-PTM, or are provided at the application layer.

For the OTA-PTM bindings Push/MBMS and Push/BCAST, the security of PTM-Push is provided by the specific bearer. PTM-Push thus relies upon MBMS [TS33.246] or BCAST [BCAST-Security] to ensure service and content protection. MBMS and BCAST define security functions supporting the relationship of specific users/clients to secured services thoughout the service lifecycle, including registration for access to services, delivery of security keys, delivery/reception of secured content, and termination of access to services.

For Push/CBS, no explicit security support is provided by the bearer. Any confidentiality requirements are the responsibility of the Push Initiator to address, e.g. through delivery of pre-encrypted content via PTM-Push, with content decryption by the target application. Any service lifecycle aspects are also the responsibility of the Push Initiator and Client Application to coordinate.

For OTA-PTM operations that occur over point-to-point bearers, e.g. Push Service Registration and unicast Push delivery as applicable to a OTA-PTM enabled service, the applicable client and server authentication mechanisms are provided by the specific point-to-point transport protocol binding. Push Clients and Push Servers MAY rely upon the authentication mechanisms provided by the underlying point-to-point transport protocol binding.

Push Clients and Push Servers MAY support requester authentication by network address.

For secure HTTP connections, Push Clients MAY support authentication of Push Servers by presence of a valid server certificate received in TLS connection setup.

For HTTP connections, Push Clients and Push servers MAY support the client authentication mechanisms defined in [SEC_CF]:

- client certificates
- Generic Bootstrap Architecture (GBA)-derived credentials used as username and password in HTTP Digest authentication, or as pre-shared secret in PSK-TLS

Authorization for use of Push services, e.g. the authorization of a Push Server to deliver Push content to a Push Client, is a deployment-specific concern. Where necessary, authorization is assumed to be based upon the pre-existence of a Push service delivery arrangement, e.g. through the Push Service Registration, and presence of valid authentication for the interfacing entity, as required.

Trust relationships between Push Clients and Push Servers are based upon the supported mechanisms for authentication and authorization, i.e. either through the facilities of the specific OTA-PTM bindings, or through explicit support by the Push Client and Push Server. Explicit mechanisms for trust establishment and termination include:

- management of the Push Whitelist
- the Push Service Registration procedure

Trust relationships between Push Clients and Push Intiators are based upon the trust each places in the Push Server via which Push content is delivered. Mechanisms for directly establishing and terminating trust between Push Client and Push Intiator are application-specific and not defined by OMA Push. As described in section 6.2.4, the Push Server can disclose its trust of the Push Intiator, and the address of the Push Intiator, via Push-OTA message headers. The Push Client, as described in section 6.2.5, can use these indications to associate specific Push messages with trusted Push Initiators.

12.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* [W-TCP]

The ***Version*** field indicates the version of SIA content type. For this specification version, its value is 2. 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-orientated push using OTA-WSP will be compatible with older PPGs (using version 0), such terminals MUST also support SIA version 0. Version 0, 1, and 2 are identical with respect to the common and WSP specific fields, except for the version number. Version 1 and 2 are identical 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 [W-TCP]. 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 [W-TCP].

The ***Contact Points*** field contains a list of PPG addresses the terminal should contact using another protocol than OTA-WSP. In the case of OTA-HTTP, the terminal should establish an active TCP connection (or connections) when contacting the PPG(s) using TO-TCP. In the case of OTA-SIP, the Push Client should register with the SIP/IP Core network (if not already registered) and send a SIP OPTIONS request to the PPG if a URI is included in the bearer address. In the case of OTA-PTM, the Push Client should initiate the Push Service Registration procedure (see section 9.1.3) with the PPG(s) at the contact address (Push Service Registration URI) for each. Each address in the field uses the *AddressType* as defined in [W-TCP].

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
2	OTA-SIP
3	OTA-PTM

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

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 SCRRULES.

A.1 Client/Terminal Features

Item	Function	Reference	Requirement
OTA-CL-C-001-M	Connectionless Push	5, 6.2.1	(WSP-CL-C-002 AND WSP-CL-C-003 AND WSP-CL-C-020 AND OTA-CL-C-002-M AND OTA-CL-C-004-O AND OTA-SEC-C-001-M AND WDP:MCF) OR OTA-CL-C-008-O
OTA-CL-C-002-M	Non-secure Port for connectionless push	6.2.1	WDP-RP-C-001
OTA-CL-C-003-O	Secure Port for WTLS for connectionless push	6.2.1	OTA-CL-C-001-M AND WDP-RP-C-002 AND WTLS:MCF AND WTLS:WTLS-C-007
OTA-CL-C-004-O	Support for a minimum of 4 SMS Bearer Type Message Concatenation	6.2.1.1	WDP-CDMA-C-001 OR WDP-GSM-C-001
OTA-CL-C-008-O	Connectionless Push using OTA-SIP	8.2.4	OTA-SIP-C-001, OTA-SIP-C-006
OTA-CO-C-001-O	Connection-orientated push	5	OTA-CO-C-002 OR OTA-CO-C-003
OTA-CO-C-002-O	Connection-orientated Push using OTA-WSP	6.2.2	(OTA-WSP-C-001 OR OTA-WSP-C-002) AND (OTA-WSP-C-003 OR OTA-WSP-C-004) AND OTA-WSP-C-005 AND OTA-SEC-C-001
OTA-CO-C-003-O	Connection-orientated Push using OTA-HTTP	7	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 OTA-SEC-C-001 AND HTTP-C-S-001
OTA-CO-C-004-O	Connection-orientated Push using OTA-SIP	8.2.5	OTA-SIP-C-001-O, OTA-SIP-C-007-O
OTA-SEC-C-001-M	Support for Push Whitelists	8.3.1	OTA-SEC-C-003-O OR OTA-SEC-C-004-O
OTA-SEC-C-002-O	Secondary Source Authentication Connectionless	8.3.3 8.3.3.2	OTA-SEC-C-007-O

OTA-SEC-C-003-O	Push Whiltelist via WAP Provisioning (APPLICATION)	8.3.1	[ProvCont]
OTA-SEC-C-004-O	Push Whitelist via Device Management Object	8.3.1	[PushMO]
OTA-SEC-C-005-O	Whiltelist via WAP Provisioning (PXADDR)	8.3.1	[ProvCont]
OTA-SEC-C-006-O	Lockout timer for SIA	8.3.2	
OTA-SEC-C-007-O	SEC/MAC content type parameters	8.3.3	
OTA-WSP-C-001-O	Connection-orientated Confirmed Push	6.2.2	WSP-CO-C-001 AND WSP-CO-C-011 AND WSP-CO-C-034
OTA-WSP-C-002-O	Connection-orientated Unconfirmed Push	6.2.2	WSP-CO-C-001 AND WSP-CO-C-010
OTA-WSP-C-003-O	Use non-secure transport service	6.2.2	
OTA-WSP-C-004-O	Use secure transport service with WTLS	6.2.2	WTLS:MCF AND WTLS: WTLS-C-007
OTA-WSP-C-005-O	SIA/SIR	8 10.2 12	OTA-SEC-C-006-O
OTA-WSP-C-006-M	Application Addressing	6.2.3	
OTA-WSP-C-007-M	Application Dispatching	6.3.1	
OTA-WSP-C-008-O	Initiator Authentication	6.2.4	
OTA-WSP-C-009-O	Bearer Selection	6.2.6	
OTA-WSP-C-010-O	Bearer Control	6.2.6	
OTA-WSP-C-011-O	Security Considerations	11	OTA-SEC-C-006-O AND OTA-SEC-C-001-M
OTA-HTTP-C-001-O	TO-TCP	7.2.4.1	TCP:MCF
OTA-HTTP-C-002-O	PO-TCP	7.2.4.1	TCP:MCF
OTA-HTTP-C-003-O	Secure (OTA-HTTP-TLS) TO-TCP	7.2.4.1	TLS:MCF
OTA-HTTP-C-004-O	Secure (OTA-HTTP-TLS) PO-TCP	7.2.4.1	TLS:MCF
OTA-HTTP-C-005-O	Registration	7.2.5 7.2.5.1	
OTA-HTTP-C-006-O	Registration Validation	7.2.5 7.2.5.2	
OTA-HTTP-C-007-O	Support Un-authenticated Terminal Identification	7.2.6 7.2.6.1	
OTA-HTTP-C-008-O	Support Un-authenticated PPG Identification	7.2.6 7.2.6.1	
OTA-HTTP-C-009-O	Support Authenticated Terminal Identification	7.2.6 7.2.6.2	
OTA-HTTP-C-010-O	Support Authenticated PPG Identification	7.2.6 7.2.6.2	
OTA-HTTP-C-011-O	Application Addressing	7.3	
OTA-HTTP-C-012-O	Content Push	7.4	
OTA-HTTP-C-013-O	Version Control	7.5	
OTA-HTTP-C-014-O	Security Considerations	11	OTA-SEC-C-006-O AND OTA-SEC-C-001-M

OTA-HTTP-C-015-O	Bearer Indication	7.6	
OTA-HTTP-C-016-O	SIA/SIR	8 10.1 12	OTA-SEC-C-006-O
OTA-HTTP-C-017-O	Support for the `X-Wap-Push-ProvURL` header	7.2.5.4	ProvCont-CB-C-001
OTA-SIP-C-001-O	OTA-SIP	8	(OTA-SIP-C-006-O OR OTA-SIP-C-007-O), OTA-SIP-C-004-M, OTA-SIP-C-005-M, OTA-SIP-C-008-M, OTA-SIP-C-009-M
OTA-SIP-C-002-O	OTA-SIP in IMS	8.2.1	OTA-SIP-C-003
OTA-SIP-C-003-M	Use of IMS ICSI for OMA Push	8.2.1	
OTA-SIP-C-004-M	Feature tag g.oma.pusheventapp	8.2.2, 8.2.3.1	
OTA-SIP-C-005-M	Registration	8.2.2, 8.2.3.1	
OTA-SIP-C-006-O	MESSAGE method	8.2.4.2	
OTA-SIP-C-007-O	INVITE/MSRP method	8.2.5.2	
OTA-SIP-C-008-M	Application Addressing	8.3	
OTA-SIP-C-009-M	Content Push	8.4.1	
OTA-SIP-C-010-O	Content Indirection	8.4.2	
OTA-SIP-C-011-O	SIA/SIR	10.2.3, 12	
OTA-SIP-C-012-O	Security Considerations	11	
OTA-PTM-C-001-O	OTA-PTM		(OTA-PTM-C-002-O OR OTA-PTM-C-003-O OR OTA-PTM-C-004-O) AND OTA-PTM-C-005-M
OTA-PTM-C-002-O	OTA-PTM binding to MBMS	9.2	
OTA-PTM-C-003-O	OTA-PTM binding to BCAST	9.3	
OTA-PTM-C-004-O	OTA-PTM binding to CBS	9.4	
OTA-PTM-C-005-M	SIA/SIR	10.4.3, 12	
OTA-PTM-C-006-O	Push Service Registration	9.1.3	
OTA-PTM-C-007-O	Security Considerations	11	

A.2 Server/PPG Features

Item	Function	Reference	Requirement
OTA-CL-S-001-M	Connectionless Push	5, 6.2.1	OTA-CL-S-002-O AND WSP-CL-S-002 AND WSP-CL-S-003 AND WSP-CL-S-020 AND WDP:MSF
OTA-CL-S-002-O	Support for a minimum of 4 SMS Bearer type Message Segmentation	6.2.1.1	WDP-CDMA-S-001 OR WDP-GSM-S-001
OTA-CL-S-003-O	Connectionless Push using OTA-SIP	8.2.4	OTA-SIP-S-001-M, OTA-SIP-S-006-O
OTA-CO-S-001-O	Connection-orientated push	5	OTA-CO-S-002 OR OTA-CO-S-003
OTA-CO-S-002-O	Connection-orientated Push using OTA-WSP	6.2.2	(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-CO-S-003-O	Connection-orientated Push using OTA-HTTP	7	OTA-HTTP-S-001-O AND OTA-HTTP-S-007-O AND OTA-HTTP-S-008-O AND OTA-HTTP-S-009-O AND OTA-HTTP-S-010-O AND OTA-HTTP-S-011-O AND OTA-HTTP-S-012-O AND OTA-HTTP-S-014-O AND OTA-HTTP-S-015-O AND HTTP-S-C-001
OTA-CO-S-004-O	Connection-orientated Push using OTA-SIP	8.2.5	OTA-SIP-S-001-M, OTA-SIP-S-007-O
OTA-SEC-S-001-O	Authentication parameter generation	8.3.3 8.3.3.1	
OTA-WSP-S-001-O	Connection-orientated Confirmed Push	6.2.2	WSP-CO-S-001 AND WSP-CO-S-011
OTA-WSP-S-002-O	Connection-orientated Unconfirmed Push	6.2.2	WSP-CO-S-001 AND WSP-CO-S-010
OTA-WSP-S-003-O	Use non-secure transport service	6.2.2	WDP-RP-S-004
OTA-WSP-S-004-O	Use secure transport service with WTLS	6.2.2	WDP-RP-S-006 AND WTLS:MSF
OTA-WSP-S-005-O	SIA/SIR	8 10.2 12	
OTA-WSP-S-006-M	Application Addressing	6.2.3	
OTA-WSP-S-007-O	Initiator Authentication	6.2.4	
OTA-WSP-S-008-O	Bearer Selection	6.2.6	
OTA-WSP-S-009-O	Bearer Control	6.2.6	
OTA-HTTP-S-001-O	TO-TCP	7.2.4.1	TCP:MSF
OTA-HTTP-S-002-O	PO-TCP	7.2.4.1	TCP:MSF
OTA-HTTP-S-003-O	Secure (OTA-HTTP-TLS) TO-TCP	7.2.4.1	TLS:MSF

OTA-HTTP-S-004-O	Secure (OTA-HTTP-TLS) PO-TCP	7.2.4.1	TLS:MSF
OTA-HTTP-S-005-O	Registration	7.2.5 7.2.5.1	
OTA-HTTP-S-006-O	Registration validation	7.2.5 7.2.5.2	
OTA-HTTP-S-007-O	Support Un-authenticated Terminal Identification	7.2.6.1	
OTA-HTTP-S-008-O	Support Authenticated Terminal Identification	7.2.6.2	
OTA-HTTP-S-009-O	Support Authenticated PPG Identification	7.2.6.2	
OTA-HTTP-S-010-O	Application Addressing	7.3	
OTA-HTTP-S-011-O	Content Push	7.4	
OTA-HTTP-S-012-O	Version Control	7.5	
OTA-HTTP-S-013-O	Security Considerations	11 8.3.3.1	
OTA-HTTP-S-014-O	Bearer Indication	7.6	
OTA-HTTP-S-015-O	SIA/SIR	8 10.1 12	
OTA-HTTP-S-016-O	Support for the `X-Wap-Push-ProvURL` header	7.2.5.4	
OTA-SIP-S-001-O	OTA-SIP		(OTA-SIP-S-006-O OR OTA-SIP-S-007-O), OTA-SIP-S-004-M, OTA-SIP-S-005-M, OTA-SIP-S-008-M, OTA-SIP-S-009-M
OTA-SIP-S-002-O	OTA-SIP in IMS		OTA-SIP-S-003-O
OTA-SIP-S-003-O	Use of IMS ICSI for OMA Push		
OTA-SIP-S-004-O	Feature tag g.oma.pusheventapp	8.2.3.2, 8.2.4.1, 8.4.3.2, 10.2.2	
OTA-SIP-S-005-O	Registration	8.2.2, 8.2.3.2	
OTA-SIP-S-006-O	MESSAGE method	8.2.4.1	
OTA-SIP-S-007-O	INVITE/MSRP method	8.2.5.1	
OTA-SIP-S-008-O	Application Addressing	8.3	
OTA-SIP-S-009-O	Content Push	8.4.1	
OTA-SIP-S-010-O	Content Indirection	8.4.2	
OTA-SIP-S-011-O	SIA/SIR	10.2.2, 12	
OTA-SIP-S-012-O	Security Considerations	11	
OTA-PTM-S-001-O	OTA-PTM		(OTA-PTM-S-002-O OR OTA-PTM-S-003-O OR OTA-PTM-S-004-O) AND OTA-PTM-S-005-M
OTA-PTM-S-002-O	OTA-PTM binding to MBMS	9.2	
OTA-PTM-S-003-O	OTA-PTM binding to BCAST	9.3	
OTA-PTM-S-004-O	OTA-PTM binding to CBS	9.4	
OTA-PTM-S-005-M	SIA/SIR	10.4.2, 12	

OTA-PTM-S-006-O	Push Channel	9.1.2	
OTA-PTM-S-007-O	Push Service Registration	9.1.3	
OTA-PTM-S-008-O	Security Considerations	11	

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version –or- No previous version within OMA

B.2 Draft/Candidate Version 2.3 History

Reference	Date	Sections	Description
Draft Versions OMA- TS-PushOTA-V2_3	22 Jun 2009	2.1, 9	Baseline for Push 2.3, based upon OMA-TS-PushOTA-V2_2-20090609-C, with agreed CRs: OMA-CD-PUSH-2009-0043R01-CR_OTA_TS_Adding_MBMS OMA-CD-PUSH-2009-0046R01-CR_OTA_PTM_over_BCAST
	12 Aug 2009	2,1, 2,2, 3,2, 3,3, 10.1, 10.2, 10.3, 10.4, 11.4	Updated for agreed CRs: OMA-CD-PUSH-2009-0051R01-CR_OTA_PTM_over_CBS OMA-CD-PUSH-2009-0064R01-CR_OTA_TS_Corrections OMA-CD-PUSH-2009-0065-CR_OTA_TS_SDP_example OMA-CD-PUSH-2009-0066R02-CR_OTA_Push_Channel OMA-CD-PUSH-2009-0071-CR_Push_OTA_Security_AI_A009
	02 Sep 2009	C, 10.1, 10.2	Updated for agreed CR: OMA-CD-PUSH-2009-0076R02-CR_Push_Channel_Registration
	28 Sep 2009	9.4, 10.1.2, 12, C	Updated for agreed CR: OMA-CD-PUSH-2009-0085-CR_A010_SIR_for_OTA_PTM OMA-CD-PUSH-2009-0084-CR_A013_delivery_notification
	30 Sep 2009	3.2, 3.3, A	Updated for agreed CR: OMA-CD-PUSH-2009-0089-CR_PushOTA_Edits_for_ETR
	07 Oct 2009	2.1, 3.2, 10.1.2, 10.2.4.1, 10.3.1, 10.3.2.1, 10.3.3.1 10.3.4, 11.1	Updated for agreed CR: OMA-CD-PUSH-2009-0096-CR_PushOTA_Edits_for_AI_A014_etc
	19 Oct 2009	2.1, 2.2, 4, 5	Updated for agreed CR: OMA-CD-PUSH-2009-0102R01-CR_PushOTA_TS_edits
	30 Nov 2009	1, 2.1, 4, 8.2.3.2, 9.1.1, 10.2.2,	Updated for agreed CR: OMA-CD-PUSH-2009-0115-CR_2.3_Push_OTA_CONRR_Updates
	21 Dec 2009	2.1, 11.4	Updated for agreed CR: OMA-CD-PUSH-2009-0128-CR_2.3_Push_OTA_CONRR_Updates_part_2
	20 Jan 2010	9.1.3 App D	Updated for agreed CR: OMA-CD-PUSH-2010-0002-CR_2.3_CONRR_SIA_IANA_Registration OMA-CD-PUSH-2010-0010-CR_2.3_CONRR_N009
	27 Jan 2010	9.1.3	Updated for agreed CR: OMA-CD-PUSH-2010-0004R01-CR_2.3_CONRR_C002.doc OMA-CD-PUSH-2010-0005-CR_2.3_CONRR_C003.doc
	24 Feb 2010	9.1.3	Updated for agreed CR: OMA-CD-PUSH-2010-0016-CR_fix_validation_Issues_Push23_Registration_1_0
Candidate Versions OMA- TS-PushOTA-V2_3	16 Mar 2010	All	Status changed to Candidate by TP: OMA-TP-2010-0106-INP_PUSH_V2_3_ERP_for_Candidate_Approval

Appendix C. Examples (Informative)

C.1 Push/MBMS Service Registration

A Client Application obtains the registration URI of a Push-enabled service. The Client Application registers with the Push Client for access to Push events, providing the Push service registration URI, and other options as its Application Profile. The Push Client registers with the Push-enabled service, and receives confirmation with a set of applicable Connection Profiles (connectivity parameters and other OMA Push options), which include an MBMS User Service Description (USD) Metadata Fragment. The Push Client recognizes the SDP as MBMS-related, and registers with the MBMS receiver as an MBMS-enabled application. The MBMS receiver completes registration with the MBMS server (BM-SC) as necessary, and subsequently Push events are delivered to the application via the established registration.

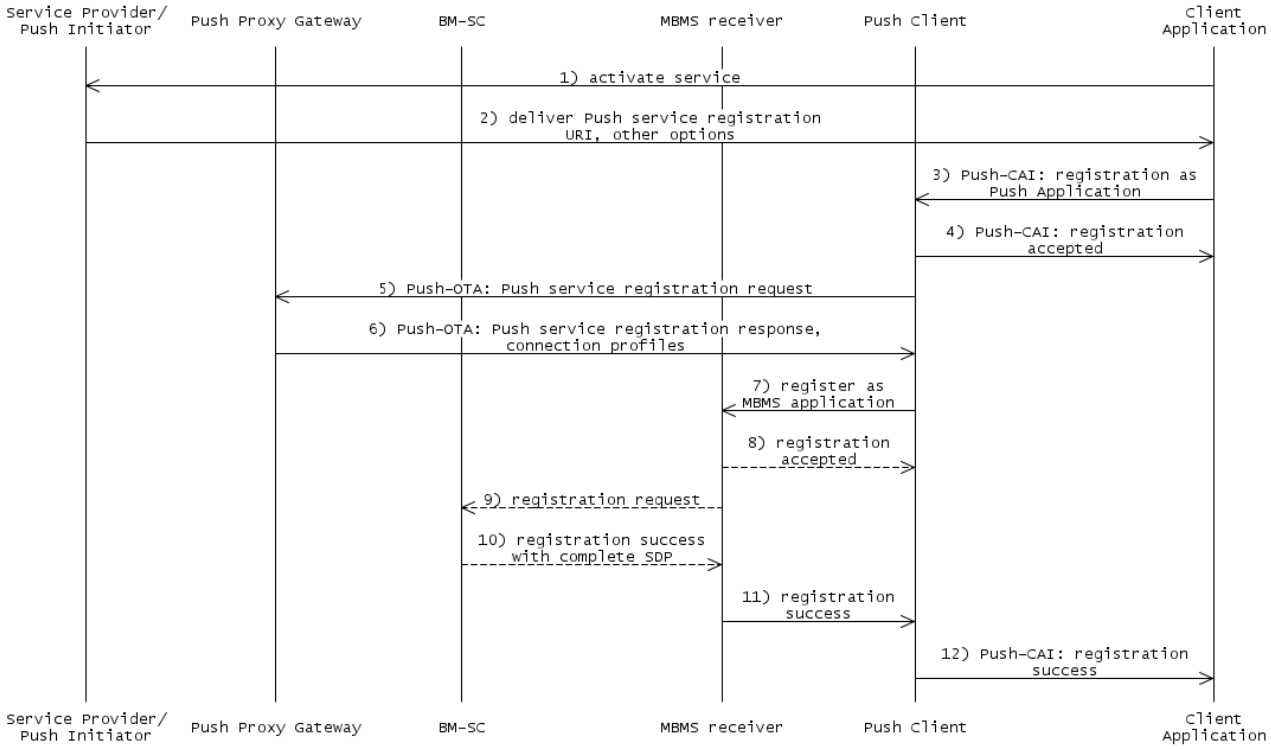


Figure 14 Push/MBMS Service Registration example

1. A Client Application activates a Push/MBMS-enabled service offered by a Service Provider.
2. The Service Provider delivers a Push service registration URI and other options as necessary for access to the Push/MBMS-enabled service.
3. Via the Push-CAI interface, the Client Application initiates registration for Push events through the Push Client, providing the Push service registration URI, its callback information (platform-dependent), and other options related to the Push-enabled service as an Application Profile. The Application Profile can include a Push Application ID (standardized or proprietary), connection profile information known by the Client Application (e.g. an SDP fragment for a specific Push/MBMS service), and other OMA Push service options as applicable (e.g. requested content types).
4. Via the Client Application callback, the Push Client indicates acceptance of registration. This means that the Push Client is taking actions to complete the registration, and the Client Application should expect a subseqent registration success event.

5. The Push Client sends a registration request to the Push service registration URI, indicating its supported capabilities and any Application Profile information as provided by the Client Application.

6. The PPG responds with further connection profile information as necessary for the Push Client to complete connection to the applicable Push bearers for the specific service. For Push/MBMS, this may include a full MBMS User Service Description or a User Service Description fragment. The PPG may take other actions as well, e.g. inform the BM-SC, Service Provider, or update its delivery context information for the Push Client to indicate that Push/MBMS is activated for the specific Push/MBMS service. This information can be used by the PPG in later decisions, e.g. whether to use unicast methods for a particular Push Client that is not Push/MBMS registered at the time of a related Push event.

7. The Push Client detects a MBMS-related connection profile in the PPG response. Via an interface exposed by the MBMS receiver, the Push Client registers as an MBMS application, providing the User Service Description, and its callback information (platform-dependent).

8. Via the Push Client callback, the MBMS receiver indicates acceptance of registration. This means that the MBMS receiver is taking actions to complete the registration, and the Push Client should expect a subseqent registration success event.

9. If the MBMS User Service Description contains a registration description, the MBMS receiver registers with the registration server at the RegistrationURL, as described in TS26346.

10. The registration server confirms the registration, and provides the remaining information necessary for the MBMS receiver to access the FLUTE channel. The MBMS receiver takes actions as necessary to open the FLUTE channel for content reception.

11. Via the Push Client callback, the MBMS receiver indicates registration success.

12. Via the Client Application callback, the Push Client indicates registration success. This means that the Push/MBMS service setup is complete, and the Client Application should expect Push events to be delivered via its callback.

C.2 Push/MBMS Message Delivery

This flow shows a Push message delivered as a new or replacement message, to a group of Push Clients via Push/MBMS.

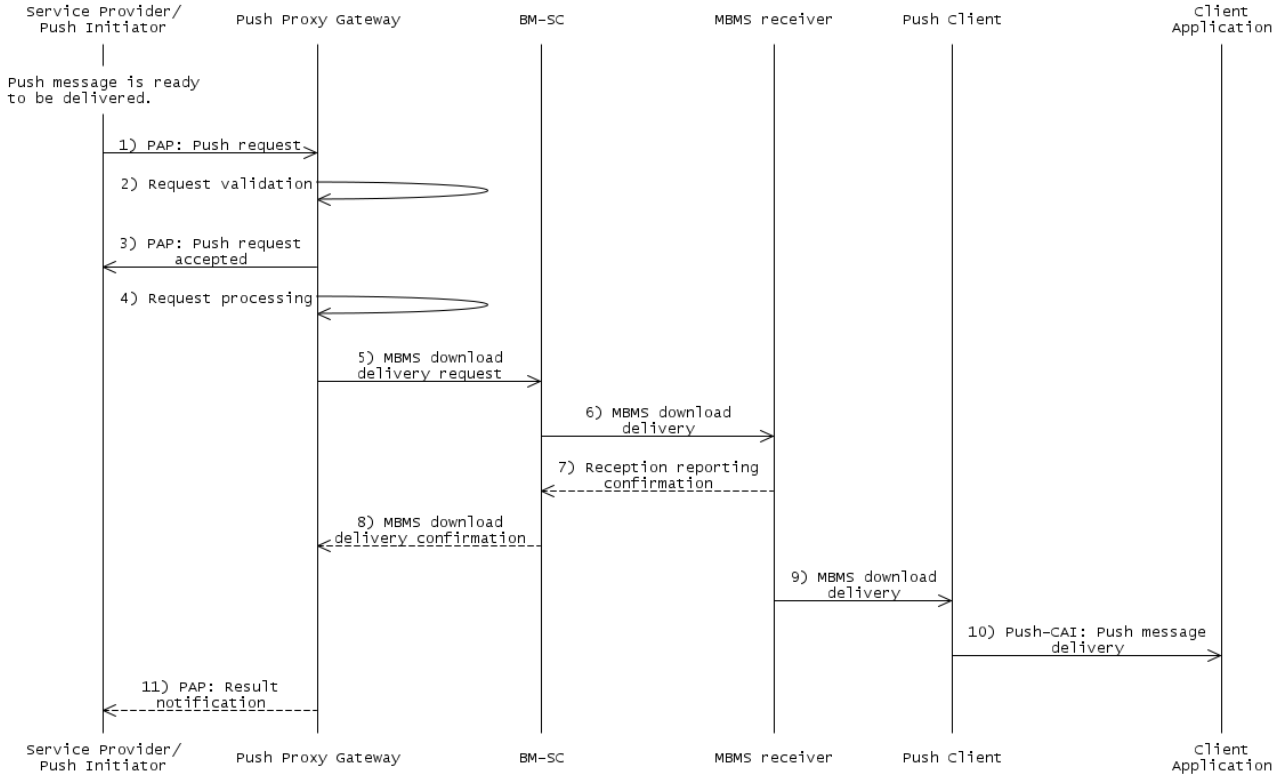


Figure 15 Push/MBMS Message Delivery example

1. Via the PAP interface [PushPAP], a Push Initiator delivers a Push request to the PPG, for delivery as a new message or replacement of a prior message. The Push Initiator indicates the target Push Application, selects MBMS as the bearer, and identifies the target user group or MBMS target address.

2. The PPG validates the request, and may accept or reject the Push request with respect to its validation or other implementation/deployment specific criteria (e.g. policies).

3. Via the PAP interface, the PPG sends a response to the Push request, indicating acceptance. If the PPG had rejected the Push request, the flow would terminate at this point.

4. For accepted Push requests, the PPG performs Push message processing which may include replacing a Push message previously submitted, queuing, mapping of the target address to the user group, obtaining group information (addresses of individual members, as applicable), selection of the Push-OTA variant for specific users (e.g. those not currently Push/MBMS enabled), transformation of the Push message in preparation for OTA transmission, etc. Transformation may include enveloping the Push message and headers in a content type application/vnd.oma.push, e.g. if any OMA-specific or application headers must be delivered.

5. Via the interface exposed by the BM-SC, the PPG requests MBMS download delivery of the Push message, indicating as necessary the specific target address.

6. The BM-SC delivers the Push message as a file via MBMS download delivery.

7. If the BM-SC provided parameters requiring reception reporting confirmation, the MBMS Receiver confirms the content reception.

8. If applicable to the interface provided by the BM-SC, the BM-SC confirms the MBMS download delivery.

9. Via the Push Client callback, the MBMS receiver delivers the MBMS downloaded content, including the content-type and file object.

10. The Push Client removes the Push message envelope if any. If the content matches the Client Application's preferences (if any), via the Client Application callback, the Push Client delivers the Push message, including the content-type, other application-related headers as received, and the Push message body.

11. If the Push Initiator had requested a result notification, the PPG will send the result notification to the Push Initiator via the PAP interface after receiving the confirmation message(s) from the BM-SC. Note the PPG may use various implementation/deployment specific criteria for determine when to send the result notification, e.g. determine success/fail based upon the number of responses received after a confirmation waiting period.

Appendix D. MIME media types

D.1 Media-Type Registration Request for application/vnd.wap.sia

This section provides the registration request, as per [RFC 2048], to be submitted to IANA.

MIME media type name : Application

MIME subtype name : Vendor Tree - vnd.wap.sia

Required parameters : none

Optional parameters : none

Encoding considerations : binary

This media type may require encoding on transports not capable of handling binary.

Security considerations :

The information contained in the media type may need privacy and integrity services but these are provided by an underling security protocol (e.g. TLS) and not by the media type itself.

As the media type represents a command to establish a data session for delivery of OMA Push events and for arbitrary application-specific purposes, the security requirements safeguarding application actions upon session establishment are the responsibility of the application to which the Session Initiation Application is delivered, or of the OMA Push Client which implements the OMA Push specifications.

The security requirements of the application/vnd.wap.sia content itself and the security implications of the use of Session Initiation Application as governed by the OMA Push enabler specifications.

Interoperability considerations :

This media type is intended for delivery to clients that implement the OMA Push enabler protocols. Client support is determined through client registration with an OMA Push Proxy Gateway (PPG) or other means via which the PPG determines the capabilities of the target client, e.g. OMA User Agent Profile.

Published specification :

The Session Initiation Application is defined in the Push Over-the-Air specification, available as part of the OMA Push enabler releases at http://www.openmobilealliance.org/Technical/current_releases.aspx.

Applications which use this media :

Mobile device browsers and any other application that conforms to the Session Initiation Application specification.

Additional information :

1. Magic number(s) : none

2. File extension(s) : none

3. Macintosh file type code : none

4. Object Identifiers: none

OMA Push - is a standard way to deliver server-initiated events (content, notifications, application data in general) to clients. OMA Push can be used to deliver any type of discrete content. It is widely supported by mobile clients in particular, and deployed as a key enabler for a variety of services standardized by the OMA, in addition to non-standardized applications and

services. The application/vnd.wap.sia media type enables establishment of data sessions for delivery of OMA Push events and for arbitrary application-specific purposes.

Person to contact for further information :

1. Name : Open Mobile Alliance, Content Delivery Working Group Chairperson

2. E-mail : technical-comments&mail.openmobilealliance.org

Intended usage : Limited Use

This MIME media type is registered for use with the OMA Push Enabler.

Author/Change controller : OMNA - Open Mobile Naming Authority,

OMA-OMNA&mail.openmobilealliance.org