Push Client - Application Interface

Candidate Version 1.0 – 09 Jun 2009

Open Mobile Alliance

OMA-TS-PushCAI-V1_0-20090609-C

Contents

1. Scope

This specification defines the Push Application Interface through which Push Applications can access OMA Push enabler services. The interface specified in this document defines functions and data formats supporting:

- registration of Push Applications with an OMA Push Client
- the subsequent delivery of Push events to the Push Application

2. References

[PushMsg] "Push Message Specification". Open Mobile Alliance™. OMA-TS-Push_Message-V2_2. URL: http://www.openmobilealliance.org/

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

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

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

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

2.1 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/

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

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

3. Terminology and Conventions

3.1 Conventions

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

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

3.2 Definitions

Application	A value-added data service provided to a Client. The application may utilise both push and pull data transfer to deliver content
Application-Level Addressing	the ability to address push content between a particular user agent on a client and push initiator on a server
Bearer Network	a network used to carry the messages of a transport-layer protocol between physical devices. Multiple bearer networks may be used over the life of a single push session.
Client	In the context of push, a client is a device (or service) that expects to receive push content from a server. In the context of pull, it is a device initiates a request to a server for content or data. See also "device".
Contact Point	address information that describes how to reach a push proxy gateway, including transport protocol address and port of the push proxy gateway.
Content	subject matter (data) stored or generated at an origin server. Content is typically displayed or interpreted by a user agent on a client. Content can both be returned in response to a user request, or be pushed directly to a client.
Content Encoding	when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.
Content Format	actual representation of content.
Device	is a network entity that is capable of sending and/or receiving packets of information and has a unique device address. A device can act as either a client or a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.
End-user	see "user"
Multicast Message	a push message containing a single address which implicitly specifies more than one OTA client address.
Push Access Protocol	a protocol used for conveying content that should be pushed to a client, and push related control information, between a Push Initiator and a Push Proxy/Gateway.
Push Framework	the entire push system. The push framework encompasses the protocols, service interfaces, and software entities that provide the means to push data to user agents in the client.
Push Initiator	the entity that originates push content and submits it to the push framework for delivery to a user agent on a client.
Push OTA Protocol	a protocol used for conveying content between a Push Proxy/Gateway and a certain user agent on a client.
Push Proxy Gateway	a proxy gateway that provides push proxy services
Push Session	A WSP session that is capable of conducting push operations.
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	a device (or service) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client. A server may initiate a connection to a client as part of a service (push).
Terminal	see "client".
Terminal-ID	an identifier that is used by a PPG to uniquely identify a terminal.
User	a user is a person who interacts with a user agent to view, hear, or otherwise use a rendered content. Also referred to as end-user.
User agent	a user agent (or content interpreter) is any software or device that interprets resources. This may include textual browsers, voice browsers, search engines, etc.

3.3 Abbreviations

ABNF	Augmented Backus-Naur Form
CPI	Capability and Preference Information
CSD	Circuit Switched Data
DNS	Domain Name Server
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
MSISDN	Mobile Station International Subscriber Directory Number
OMA	Open Mobile Alliance
OTA	Over The Air
OTA-HTTP	(Push) OTA over HTTP
OTA-SIP	(Push) OTA over SIP
OTA-HTTP-TLS	OTA-HTTP over TLS
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
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

The OMA Push architecture enables a Push Client to register for Push-based services on behalf of Push applications, and to subsequently deliver Push events to the Push application. The Push Client may use various over-the-air (OTA) protocols and data bearers in providing Push service.

Historically, Push Clients have been implemented as embedded software functions in mobile devices, with pre-defined support for specific Push applications, e.g. browsers, Multimedia Messaging clients, Instant Messaging clients, etc. The Push Client typically registers as needed on behalf of these pre-defined Push applications, and includes its capabilities on their behalf in the vendor-published User Agent Profile for the device. While this approach has supported well the growth of Push-based services for typical embedded Push applications, it does not support the dynamic enhancement of devices for new Push applications and services, e.g. through installation of downloadable applications and OMA service enabler clients.

This specification enables Push Clients to optionally offer a means for Push applications to register for Push services with application-specified options for the Push service, including:

- Providing an application name, which can be a unique identifier usable by OMA Push entities in service matching
- A Push Application ID, which associates the application with a OMNA-registered Push service type
- A specified application event handler for reception of Push events
- A selection of Push Servers (e.g. Push Proxy Gateways or Push Application Servers) from which the Push Client should request service and expect Push events
- Push protocols and data bearers to use for Push Client registration and Push event reception
- Push event addressing at the Push Client, via which the Push Client can be directed to receive Push events at standard or alternate ports for specific bearers
- A set of content types which the Push application is compatible with, which may be used by the Push Client to negotiate capabilities with a Push Server

This registration and event reception interface enables the dynamic extension of the OMA Push enabler to new applications and OMA service enablers as described above.

4.1 Version 1.0

Version 1.0 includes all of the functionality described in section 4.

5. Push Client - Application Interface (Push-CAI)

The term Push-CAI is used in this specification to refer to a generic interface defined as a set of functions and related data formats enabling the high-level capabilities described in section 4, which comprise overall:

- Push application registration with the Push Client, and subsequent deregistration if required
- Push Client delivery of Push events to the Push application

The Push-CAI is defined in an implementation-agnostic manner, since the implementation of the Push-CAI in specific devices will depend upon the application programming environments and runtime environments supported by the devices. Thus definition of a specific application programming interface (API) is out of scope for this specification. However to support the basic set of interface functionality called for in this specification, the Push-CAI makes the following assumptions about the interaction capabilities of the Push Client and Push applications:

- The Push Client supports some means of being invoked for registration of the Push application. This can be for example through a callable function or a service access point to which interface messages can be directed.
- The Push application, through the execution environment, supports some means of being invoked for Push event delivery from the Push Client. Similar to the Push Client registration function, the Push application may provide this through a callback function or a service access point to which interface messages can be directed.

5.1 Push Application Registration

Push applications register with the Push Client through delivery of an Application Registration request to the Push Client, and receiving an Application Registration response. The Application Registration may be successful or unsuccessful depending upon the Push Client's ability to handle the Push application's specified options.

The Push application must include the following parameters in the Application Registration request:

- pushApplicationID (string): an [OMNA] registered Push Application ID to be associated with the Push application.

The Push application may include the following parameters as options in the Application Registration request:

- applicationID (string): "name" of the application, recommended as a unique identifier in the form of a URI registered by the Push application developer.
- eventHandler (string): handler for incoming Push events, defined per the capabilities of the device execution environment and Push application preferences. May be unspecified if the device execution environment provides default mechanisms for Push event delivery to the Push application.
- whitelist (structure): set of addresses from which Push events should be accepted per [OMA Push22]. If unspecified, the Push Client default whitelist (if any) is assumed to be compatible with the Push application.
- connectionProfile (structure): one or more set of options for Push event delivery, including
    - protocol (string): "OTA-WSP", "OTA-HTTP", or "OTA-SIP". If unspecified, the default supported protocols are assumed to be compatible with the Push application.
    - bearerType (string): a bearer type to be used with the requested protocol
    - port (string): a specific port number appropriate for the protocol and bearerType. If unspecified, the default/standard port should be used. Note that the mechanisms for coordination of non-standard port usage are unspecified, e.g. are assumed to occur at the application layer.
- Accept (string): a set of MIME types of Push events which the Push application specifically requests subscription to.

The Push Client MUST reject an Application Registration indicating an already-registered Push Application ID, unless:

- the Application Registration is received from the same Push application, or
- the Push Client supports a mechanism to ensure uniqueness in Push application registration with the Push Server, and delivery of Push events intended for the specific Push application, e.g.
    - The OTA-SIP protocol is used. OTA-SIP provides a means to uniquely identify distinct Push applications supporting the same Push Application ID, via a media feature tag indentifying the specific application instance, e.g.
        - in IMS, via the IMS Application Reference Identifier (IARI), as described in [Push-OTA].
        - in non-IMS SIP networks, using the same feature tag name
    - The Push application is registered for event reception at a unique bearer port, by request of the Push application.

The Push Client MUST reject an Application Registration with an option that is not supported or not currently available.

If it accepts the Application Registration request, the Push Client MUST:

- respond to the Push application with an Application Registration response indicating registration success, and
- register with the Push Server (e.g. Push Proxy Gateway or Push Application Server) for the Push application if the applicable Push protocol supports explicit registration.

In registering with the Push Server, the Push Client MUST:

- use the applicable bearer for the Application Registration
- indicate the new Push Application ID
    - for OTA-HTTP, via the X-Wap-Push-Accept-AppID header provided in the terminal's CPI
    - for OTA-SIP, via inclusion as a Push Resource Indentifier in
        - a SIP REGISTER request sent to update the registration with the SIP/IP Core
        - a SIP OPTIONS request sent to update the capabilities requested from the Push Server
- indicate the specified MIME types, if any, via inclusion in
    - for OTA-HTTP, the X-Wap-Push-Accept header provided in the terminal's CPI
    - for OTA-SIP, the SIP OPTIONS request sent to update the capabilities requested from the Push Server

If the registration fails for any reason, the Push Client MUST respond to the Push application with an Application Registration response indicating an applicable failure reason,

If it accepts an Application Registration containing a whitelist parameter, the Push Client MUST add the specified Push Servers to its current whitelist, for the specific Push application.

The Push Client MUST retain the details of the registration request for future reference in a deregistration process if required.

5.2 Push Application Deregistration

Push applications deregister with the Push Client through delivery of an Application Deregistration request to the Push Client, and receiving an Application Deregistration response. The Application Deregistration may be successful or unsuccessful depending upon validation of the request by the Push Client.

The Push application must include the following parameters in the Application Deregistration request:

- pushApplicationID (string): an [OMNA] registered Push Application ID that is associated with the Push application.

The Push Client MUST reject an Application Deregistration request if it determines that the requesting Push application is not currently registered for events related to the Push Application ID.

If it accepts the Application Deregistration request, the Push Client MUST deregister with the Push Server (e.g. Push Proxy Gateway or Push Application Server) for the Push application if the applicable Push protocol supports explicit deregistration.

In deregistering with the Push Server, the Push Client MUST:

- use the applicable bearer for the Application Registration
- deliver an updated set of supported Push Application ID's , removing Push Application ID associated with the deregistered Push application
    - for OTA-HTTP, via exclusion in the X-Wap-Push-Accept-AppID header provided in the terminal's CPI
    - for OTA-SIP, via exclusion as a Push Resource Indentifier in
        - a SIP REGISTER request sent to update the registration with the SIP/IP Core
        - a SIP OPTIONS request sent to update the capabilities requested from the Push Server
- deliver an updated set of supported MIME types, if any, removing any MIME types that were solely associated wth the deregistered Push application
    - for OTA-HTTP, in the X-Wap-Push-Accept header provided in the terminal's CPI
    - for OTA-SIP, in a SIP OPTIONS request sent to update the capabilities requested from the Push Server

If the registration had included an update to the Push whilelist, the Push Client MUST remove from the Push whitelist any Push Servers that were solely associated with the deregistered Push application.

If the Push Client accepts the Application Deregistration request and successfully deregisters with the Push Server (if required), the Push Client MUST respond to the Push application with an Application Deregistration response indicating deregistration success,

If the deregistration fails for any reason, the Push Client MUST respond to the Push application with an Application Deregistration response indicating an applicable failure reason,

5.3 Push Event Delivery

Push Clients deliver Push events to Push applications upon reception of the Push event from Push Server.

The Push Client MUST include the following parameters in the Push event:

- content: the Push content received in the Push event from the Push Server
- content-type: the MIME type of the Push content

6. Security Considerations

Push Clients or the device runtime environment in which they execute MAY enforce unspecified security policies on Push registration requests. Unless such security policies are enforced, the Push Client MUST assume that the Push application is trusted and authorized to request registration with the OMA Push enabler. Note that further security policies may be applied by the network, e.g. Push Server.

Appendix A. Change History (Informative)

A.1 Approved Version History

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

A.2 Draft/Candidate Version 2.2 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-PushAPI-V2_2	11 Nov 2008	All	Initial draft
	12 Jan 2009	5, 5.1, 5.2, 6, B	OMA-CD-PUSH-2008-0187R01-CR_PushCAI_Update
	20 Apr 2009	2, 2.1, 5.4	Per agreed CONRR changes: OMA-CD-PUSH-2009-0027-CR_CONRR_PushCAI_resolutions
Candidate Version: OMA-TS-PushAPI-V2_2	09 Jun 2009	All	Status changed to Candidate by TP: OMA-TP-2009-0200R01-INP_Push_V2_2_ERP_for_Candidate_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for Push Client

Item	Function	Reference	Requirement
CAI-C-001-M	Push Application Registration	5.1	
CAI-C-002-M	Push Application Deregistration	5.2	
CAI-C-003-M	Push Event Delivery	5.3	
CAI-C-004-O	Security Considerations	6	