WAP Push OTA Protocol

Version 08-Nov-1999

Wireless Application Protocol
Push OTA Protocol Specification

Contents

1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and providing new services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation, and fast/flexible service creation, WAP defines a set of protocols in transport, session and application layers. For additional information on the WAP architecture, refer to "*Wireless Application Protocol Architecture Specification"* [WAP].

This specification defines the Over the Air protocol for delivery of content to a WAP client from a WAP server, referred to as Push OTA protocol. The protocol specified in this document is an application layer protocol built on top of WSP layer, as specified in " *Wireless Session Protocol Specification*" [WSP].

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

- means for delivery of push contents to WAP clients.
- means for facilitating server initiated asychronous push.
- means for facilitating application addressing.
- means for exchange of push control information over the air.
- means for facilitating bearer selection and controls.
- means for facilitating authentication of a push initiator.

2. Document Status

This document is available online in the following formats:

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

2.1 Copyright Notice

2.2 Errata

Known problems associated with this document are published at http://www.wapforum.org/.

2.3 Comments

Comments regarding this document can be submitted to the WAP Forum in the manner published at http://www.wapforum.org/.

3. References

3.1 Normative References

[WDP] " Wireless Datagram Protocol ", WAP Forum, 05-Nov-1999,
URL:http://www.wapforum.org

[WSP] "Wireless Session Protocol", WAP Forum, 05-Nov-1999.
URL: http://www.wapforum.org

[WTLS] "Wireless Transport Layer Security Protocol", WAP Forum, 05-Nov-1999.
URL:http://www.wapforum.org

[PushMsg] “Push Message Specification”, WAP Forum, 16-August-1999.
URL: http://www.wapforum.org

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

[ABNF] “Augmented BNF for Syntax Specification: ABNF”, D. Crocker, et al. November 1997,
URL: http://www.ietf.org/rfc/rfc2234.txt

3.2 Informative References

[WAE] "Wireless Application Environment Specification", WAP Forum, 04-Nov-1999.
URL: http://www.wapforum.org

[WAP] “Wireless Application Protocol Architecture Specification”, WAP Forum, 30-Apr-1998.
URL: http://www.wapforum.org

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

4. Definitions and Abbreviations

4.1 Definitions

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

Application - A value-added data service provided to a WAP Client. The application may utilise both push and pull data transfer to deliver content

Application-Level Addressing - the ability to address push content between a particular user agent on a WAP client and push initiator on a server.

Bearer Network - a network used to carry the messages of a transport-layer protocol between physical devices. Multiple bearer networks may be used over the life of a single push session.

Client – in the context of push, a client is a device (or service) that expects to receive push content from a server. In the context of pull a client, it is a device initiates a request to a server for content or data. See also “device”.

Contact Point – address information that describes how to reach a push proxy gateway, including transport protocol addres and port of the push proxy gateway.

Content - subject matter (data) stored or generated at an origin server. Content is typically displayed or interpreted by a user agent on a client. Content can both be returned in response to a user request, or being pushed directly to a client.

Content Encoding - when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.

Content Format – actual representation of content.

Context – an execution space where variables, state and content are handled within a well-defined boundary.

Device – is a network entity that is capable of sending and/or receiving packets of information and has a unique device address. A device can act as either a client or a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.

End-user - see “user”

Extensible Markup Language - is a World Wide Web Consortium (W3C) recommended standard for Internet mark-up languages, of which WML is one such language. XML is a restricted subset of SGML.

Multicast Message - a push message containing a single OTA client address which implicitly specifies more than OTA client address.

Push Access Protocol - a protocol used for conveying content that should be pushed to a client, and push related control information, between a Push Initiator and a Push Proxy/Gateway.

Push Framework- - the entire WAP push system. The push framework encompasses the protocols, service interfaces, and software entities that provide the means to push data to user agents in the WAP client.

Push Initiator - the entity that originates push content and submits it to the push framework for delivery to a user agent on a client.

Push OTA Protocol - a protocol used for conveying content between a Push Proxy/Gateway and a certain user agent on a client.

Push Proxy Gateway - a proxy gateway that provides push proxy services.

Push Session - A WSP session that is capable of conducting push operations.

Server - a device (or service) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client. A server may initiate a connection to a client as part of a service (push).

User - a user is a person who interacts with a user agent to view, hear, or otherwise use a rendered content. Also referred to as end-user.

User agent - a user agent (or content interpreter) is any software or device that interprets resources. This may include textual browsers, voice browsers, search engines, etc.

XML – see *Extensible Markup Language*

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

CPI	Capability and Preference Information
DNS	Domain Name Server
DTD	Document Type Definition
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
OTA	Over The Air
PAP	Push Access Protocol
PI	Push Initiator
PPG	Push Proxy Gateway
QOS	Quality of Service
RDF	Resource Description Framework
RFC	Request For Comments
SGML	Standard Generalized Markup Language
SI	Service Indication
SIA	Session Initiation Application
SIR	Session Initiation Request
SL	Service Loading
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Co-ordinated
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WBXML	WAP Binary XML
WINA	WAP Interim Naming Authority
WTLS	Wireless Transport Layer Security
XML	Extensible Mark-up Language

5. Introduction

The architecture consists of a distributed client/server application, with a server residing in the *push proxy gateway* or a *push initiator*, and a client residing in the mobile device. The server and client communicate using the Push OTA protocol which utilises WSP Session [WSP] services.

Connection-oriented push requires that a push session is established before the push content can be delivered. The connection-oriented push includes both confirmed and unconfirmed push. A push session for a connection-oriented push can be shared among multiple client applications. Each one of two registered WDP ports for connectionless push can also be shared among multiple client applications. A client application is identified by application ID.

A server is able to initiate push sessions with the client. To accomplish this, the server uses the connectionless push on a registered WDP port on a client to request the client to initiate a push session. This server initiation for push sessions uses SIA.

Session oriented push generally requires client devices with two way bearer services, but it is possible that a push session can be pre-planted between a client and a server so that connection-oriented unconfirmed and connectionless push might be conducted on a device with only one way bearer services.

The overall push architecture is outlined in Figure 1.

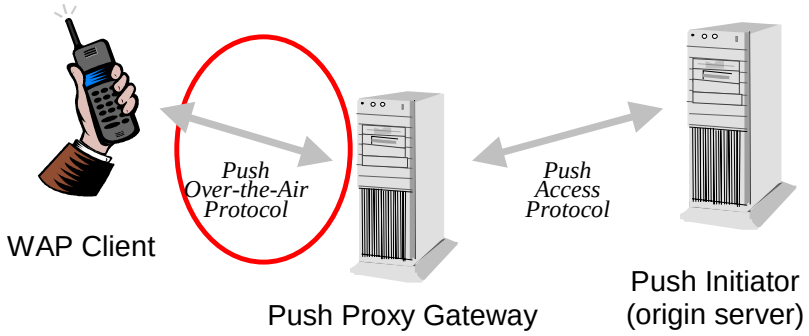

Figure 1: WAP Push Architecture

6. Service Primitive Definition

6.1 Notations

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

6.2 Service Primitive Overview

Push OTA primitives include the push operational primitives and the push management primitives. While the push operational primitives are used to deliver content from a server to a client, the push management primitives are used to establish and manage the push session.

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

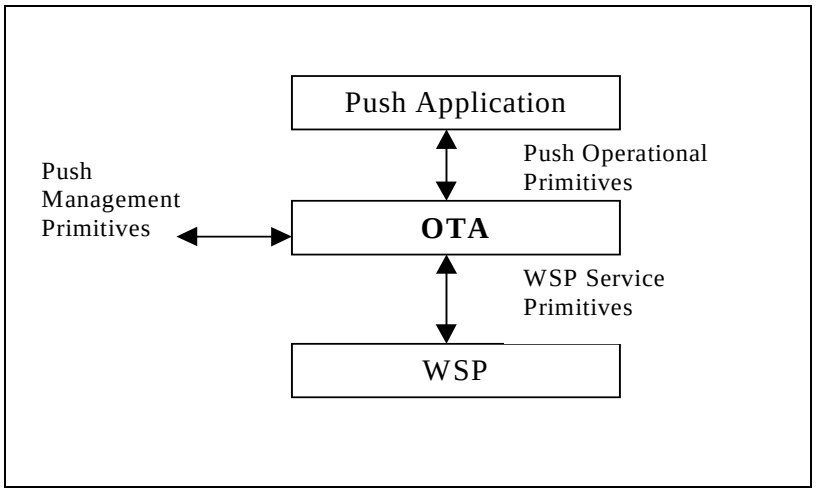

Figure 2: Illustration of Layer to Layer Communication

6.3 Push Operational Primitives

6.3.1 Po-Push

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

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

Push Body	O	C(=)

Push Headers are defined in [PushMsg].

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

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

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

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

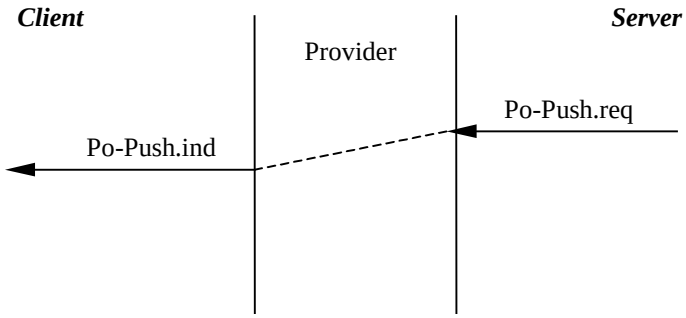

Figure 3: Unconfirmed Push

6.3.2 Po-ConfirmedPush

This primitive is used to send information from the server in a confirmed manner on a push session using the connection oriented service. It is the service user (e.g., client push application) that confirms the push by invoking Po-ConfirmedPush.res primitive when the service user takes responsibility for the push message. If the service user can not take responsibility for the push message, it MUST abort the push operation by invoking the Po-PushAbort.req primitive (6.3.3). The service provider MAY abort the push on behalf of the service user at its discretion (e.g., if the service user does not respond).

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

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

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

Push Headers are defined in [PushMsg].

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

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

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

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

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

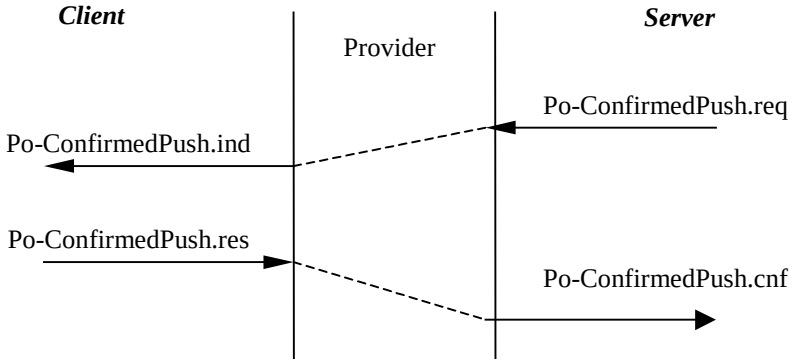

Figure 4: Confirmed Data Push

6.3.3 Po-PushAbort

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

Name	Description
USERREQ	aborted without specific causes
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 messsage is discarded due to resource shortage
USERDCU	aborted because the content type can not be processed

6.3.4 Po-Unit-Push

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

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

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

Server Address identifies the originator of the push.

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

Push Headers are defined in [PushMsg].

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

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

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

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

6.4 Push Management Primitives

6.4.1 Pom-Connect

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

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	–	–	–	–
Negotiated Capabilities	–	–	O	C(=)
Accept Application	O	C(=)	–	–
Bearer Indication	O	C(=)	–	–

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

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

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

Accept Application provides a list of *Application Id*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 can not 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].

6.4.2 Pom-Suspend

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

6.4.3 Pom-Resume

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

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

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

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

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

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

6.4.4 Pom-Disconnect

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

6.4.5 Pom-SessionRequest

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

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

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

Server Address identifies the address of the server.

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

```
Content-Type: application/vnd.wap.sia
X-Wap-Application-Id: <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 9.2.1.

7. Protocol Description

Push OTA is a protocol layer that resides on the top of the WSP session layer and provides WAP push support for either *connectionless* or *connection-oriented* mode [WSP]. Push OTA is a thin protocol layer and stateless.

7.1 Connectionless Push

The connectionless push must be performed through WSP S-Unit-Push [WSP], which is one of WSP connectionless session service primitives. Two registered WDP ports, 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]. Furthermore, the client implementation must take the responsibility to establish the secure WTLS connection between its well-known WDP secure port and the designated ports of all PPGs which are expected to send secure pushes.

7.2 Connection-Oriented Push

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

The push session can use either secure or non-secure transport services. 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.

7.3 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 port in the client, the Push OTA layer of the client is responsible to further deliver the push content to the application as identified by the Application Id. For the connection-oriented push, the push content is first delivered on the push session, the Push OTA 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 WML User Agent for the connectionless push, and also for the connection-oriented push unless the default is negotiated in the push session establishment.

7.4 Initiator Authentication

The push initiator may be authenticated through the use of the *Authenticated* flag and *Initiator URI*. The model of authentication is based the transitive trust established between a PPG and the client. The PPG MAY authenticate the push initiator on behalf of a client and MAY inform the client if a push initiator is authenticated.

The client may determine if the push initiator is trusted by comparing its list of trusted initiator URI's with the authenticated initiator URI in the push message.

7.5 Trusted Content

The Push Proxy Gateway MAY use 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.

7.6 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.

8. Protocol Operations

8.1 Application Dispatching

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

8.2 Server Initiated Session

Because delivery of push content to a client may be asynchronous, it is possible that no session exists or the session on the preferred network bearer does not exist when push is requested. This mechanism enables a server to request a client to establish a push session on a given bearer and for a given contact point for connection-oriented push.

8.2.1 Session Initiation Application

Session Initiation Application (*SIA*) is an application that can be invoked over a registered connectionless push port in the client or in a push session. It handles the server initiated session request. SIA ensures that the requested application in the client establishes a push session. SIA MUST be supported in all the clients that implement the connection-oriented push.

8.2.2 Server Procedure

When a server wishes to establish a push session with a client for a given bearer and a given contact point, it sends a session request to the SIA in that client. If the server wishes the client to establish multiple push sessions, it sends a list of contact points.

The session request is a content type (9.2.1), which is pushed to a connectionless push port in the client, and contains the following information:

- A list of Application Id's that need to have a push session;
- A list of contact points for the server;

8.2.3 Client Procedure

Using the list of contact points for the server and the list of the Application Ids, the client will create a new push session with the server. The client may also create multiple new push sessions, if applicable. It is the responsibility of the client to clean up the stale push sessions, if any. For each contact point, when a session is established, the provided Accept-Application list SHOULD NOT include Application Ids that do not exist in the client. However, the client MAY indicate (eg, due to privacy concerns) that a session accepts any Application Id..

The client must ensure that the WTLS secure connection exists before it creates the new push session, if the secure transport service is required. If the WTLS secure connection does not exist, the client must establish a WTLS secure connection prior to creating the new push session.

8.2.4 Security Considerations

To protect against denial of service attacks, the client SHOULD implement a lockout timer. If the client receives any additional session requests during the lockout interval, it should defer processing or may discard them until the timer expires. If a push session is successfully established, the lockout timer SHOULD be cancelled, if applicable. The value of the lockout timer interval is implementation specific.

To protect against spoofing, the client SHOULD validate the session request by comparing the source address the PDU that carries session request with the pre-existing list of contact points for servers. The client SHOULD ignore the session request if the validation fails.

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

9. Protocol Data Unit Definition

This section describes the protocol data units used for the Push OTA operations and OTA services.

9.1 Header Based Protocol Data Unit

The header definition rules in this sub-section follows the rules in [HTTP]. WSP compact encoding rules MUST be used to encode them for over the air efficiency.

9.1.1 Accept-Application

```
Accept-Application = “Accept-Application” “:” app-ranges
App-ranges = ( #app-id | “*” ) [PushMsg]
   ; * means any application id.
```

9.1.2 Bearer-Indication

```
Bearer-Indication = “Bearer-Indication” “:” bearer-type
Bearer-type = 2HEX              ;as defined in by WDP.
```

9.1.3 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.
; BIT is defined in [ABNF].
```

9.2 Content Based Protocol Data Unit

9.2.1 SIA content

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

Field Name	**Type**
Version	Uint8
AppIdListLen	Uintvar; number of octets for Application Id List field
Application Id List	AppIdListLen octets
ContactPointsLen	Uintvar; number of octets for Contact Points field
Contact Points	ContactPointsLen octets

The *Version* indicates the version of SIA content type. It is an 8 bit unsigned integer. For this specification, its value is 0. The future version of SIA should only add new fields at the end of this content type, if the new fields are needed, to ensure maximum backward compatibility.

AppIdListLen and *ContactPointsLen* indicate the length of the following field. Each length is encoded using the variable-length *Uintvar* integer format defined in WSP specification [WSP].

The *Application Id List* uses the same encoding as the `Accept-Application` header when it is encoded as the compact encoding format of WSP. SIA SHOULD ensure that each application as identified by the Application Id in the list establishes a push session for all the contact points.

The Contact Points contains a list of server addresses the client should contact to establish a push session. Each address in the field uses the *Address Type* as defined in WSP specification [WSP].

10. Static Conformance Requirements

This static conformance clause defines a minimum set of features that should be implemented to support OTA protocol. A feature can be optional (O), mandatory (M) or conditional (C (<condition>)). If optional/conditional features have labels (O.<n> or C.<n>), support of at least one in the group of options labelled by the same number is required.

10.1 Client Features

Item	Functionality	Reference	Status
OTAC_010	Connectionless Push	7.1	M
OTAC_011	Non-secure Port for connectionless push	7.1	M
OTAC_012	Secure Port for WTLS for connectionless push	7.1	O
OTAC_020	Connection-Oriented Push	7.2	O
OTAC-021	Confirmed Push	7.2	C.2 (OTAC_020)
OTAC-022	Unconfirmed Push	7.2	C.2 (OTAC_020)
OTAC_023	Use non-secure transport service	7.2	C.1 (OTAC_020)
OTAC_024	Use secure transport service with WTLS	7.2	C.1 (OTAC_020)
OTAC_031	Session Initiation Application	8.2.1	C (OTAC_020)
OTAC_040	Application Addressing	7.3	M
OTAC_041	Application Dispatching	8.1	M
OTAC_050	Initiator Authentication	7.4	O
OTAC_070	Bearer Selection	7.6	O
OTAC_080	Bearer Control	7.6	O
OTAC_090	Security Considerations	8.2.4	O

10.2 Server Features

Item	Functionality	Reference	Status
OTAS_010	Connectionless Push	7.1	M

Item	Functionality	Reference	Status
OTAS_011	Non-secure Port for connectionless push	7.1	M
OTAS_012	Secure Port for WTLS for connectionless push	7.1	O
OTAS_020	Connection-Oriented Push	7.2	O
OTAS-021	Confirmed Push	7.2	C.2 (OTAS_020)
OTAS-022	Unconfirmed Push	7.2	C.2 (OTAS_020)
OTAS_023	Use non-secure transport service	7.2	C.1 (OTAS_020)
OTAS_024	Use secure transport service with WTLS	7.2	C.1 (OTAS_020)
OTAS_031	Server Initiated Session	8.2	C (OTAS_020)
OTAS_040	Application Addressing	7.3	M
OTAS_041	Application Dispatching	8.1	M
OTAS_050	Initiator Authentication	7.4	O
OTAS_070	Bearer Selection	7.6	O
OTAS_080	Bearer Control	7.6	O