Device Management WSP Binding

Candidate Version 1.3 – 25 May 2010

Open Mobile Alliance

OMA-TS-DM_WSPBinding-V1_3-20100525-C

Contents

1. SCOPE 4
2. REFERENCES 5
2.1 NORMATIVE REFERENCES 5
2.2 INFORMATIVE REFERENCES 5
3. TERMINOLOGY AND CONVENTIONS 6
3.1 CONVENTIONS 6
3.2 DEFINITIONS 6
3.3 ABBREVIATIONS 6
4. INTRODUCTION 7
5. WSP MAPPING TO DM 8
5.1 MULTIPLE MESSAGES PER PACKAGE 8
5.2 MIME HEADER TYPE REQUIREMENT 8
5.3 CONNECTION ORIENTED SESSION 8
5.3.1 Session establishment, S-Connect 8
5.3.2 Exchanging DM Messages 9
5.3.3 Temporarily suspending the session, S-Suspend and S-Resume 10
5.3.4 Session close-down, S-Disconnect 10
5.4 CONNECTIONLESS SERVICE 10
5.5 PUSHING DATA FROM THE SERVER TO THE CLIENT 11
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 13
A.1 APPROVED VERSION HISTORY 13
A.2 DRAFT/CANDIDATE VERSION 1.3 HISTORY 13
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 14
B.1 CLIENT FEATURES 14
B.1.1 Client WSP Method Requirements 14
B.2 SERVER FEATURES 14
B.2.1 Server WSP Method Requirements 14

Figures

Figure 1: Session Establishment 8
Figure 2: MethodInvoke using HTTP POST 9
Figure 3: Connectionless Unit_MethodInvoke using HTTP POST 10
Figure 4: Push Model 11
Figure 5: Push scenario 12

1. Scope

DM Client

This document describes the WSP Binding for carrying DM messages based on DM representation [DMREPRO].

2. References

2.1 Normative References

[DMNotification] "DM Notification", Open Mobile Alliance™, OMA-TS-DM_Notification-V1_3, URL:http://www.openmobilealliance.org/

[DMREPRO] "DM Representation Protocol", Open Mobile Alliance™, OMA-TS-DM_RepPro-V1_3, URL:http://www.openmobilealliance.org/

[HTTPBIND] "SyncML HTTP Binding Specification", Open Mobile Alliance™, OMA-TS-SyncML_HTTPBinding-V1_3, URL:http://www.openmobilealliance.org/

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.9, Open Mobile Alliance™, OMA-IOP-Process-V1_9, URL:http://www.openmobilealliance.org/

[PUSHARCH] "Push Architectural Overview", WAP Forum, URL:http://www1.wapforum.org/tech/terms.asp?doc=WAP-250-PushArchOverview-200010703-a.pdf

[PUSHOTA] "Push Over The Air", Open Mobile Alliance™,OMA_TS-PushOTA-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

[RFC2616] "Hypertext Transfer Protocol – HTTP/1.1", IETF, RFC 2616, URL:http://www.ietf.org/rfc/rfc2616.txt

[WDP] "Wireless Datagram Protocol Specification", Open Mobile Alliance™, WAP-259-WDP-20010614-a, URL:http://www.openmobilealliance.org/Technical/wapindex.aspx

[WSP] "Wireless Session Protocol specification", Open Mobile Alliance™, WAP-230-WSP-20010705-a, URL:http://www.openmobilealliance.org/Technical/wapindex.aspx

[WTP] "Wireless Transaction Protocol Specification", Open Mobile Alliance™, WAP-224-WTP-20010710-a, URL:http://www.openmobilealliance.org/Technical/wapindex.aspx

2.2 Informative References

None.

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.

Any reference to components of the SyncML DTD or XML snippets is specified in this `typeface`.

3.2 Definitions

Connectionless Session Service	Connectionless session service is an unreliable session service. In this mode, only the request primitive is available to service users, and only the indication primitive is available to the service provider.
Connection-Mode Session Service	Connection-mode session service is a reliable session service. In this mode, both request and response primitives are available to service users, and both indication and confirm primitives are available to the service provider.
Content	The entity body sent with a request or response is referred to as content. It is encoded in a format and encoding defined by the entity-header fields.
Content Negotiation	Content negotiation is the mechanism the server uses to select the appropriate type and encoding of content when servicing a request. The type and encoding of content in any response can be negotiated. Content negotiation allows a server application to decide whether a client can support a certain form of content.
Entity	An entity is the information transferred as the payload of a request or response. An entity consists of meta-information in the form of entity-header fields and content in the form of an entity-body.
Header	A header contains meta-information. Specifically, a session header contains general information about a session that remains constant over the lifetime of a session; an entity-header contains meta-information about a particular request, response or entity body.
Loader	Entity that implements the HTTP protocol. The loader is the interface between the WSP layer and the user application.

3.3 Abbreviations

OMA	Open Mobile Alliance
PPG	Push Proxy Gateway [PUSHOTA]
WSP	Wireless Session Protocol [WSP]

4. Introduction

This document describes how to use the DM over WSP (WAP). The document uses the primitives and methods defined in the WAP Forum WSP specification as of WAP June 2000 Conformance Release.

The document describes the use of the WSP layer.

WAP [WSP] defines both a connection oriented and a connection less services for data exchange. Furthermore, it defines a server originated data-push model.

Note that the WAP specification does not specify the Loader, i.e. the interfaces to the different layers in the protocol, only the messages being used for communication in the actual layers. How the Loader is specified is up to the client (or server) vendor. The Loader interface is illustrated in the following by the messages going back and forth between the DM Client and the Loader.

The Session layer protocol family in the WAP architecture is called the Wireless Session Protocol, WSP. WSP provides the upper-level application layer of WAP with a consistent interface for two session services. The first is a connection-mode service that operates above a transaction layer protocol WTP, and the second is a connectionless service that operates above a secure or non-secure datagram transport service. For more information on the transaction and transport services, please refer to [WTP] "*Wireless Application Protocol: Wireless Transaction Protocol Specificatio*n" and [WDP] [WDP]"*Wireless Application Protocol: Wireless Datagram Protocol Specificatio*n". WSP provides HTTP 1.1 functionality and incorporates new features such as long-lived sessions, a common facility for data push, capability negotiation and session suspend/resume. The protocols in the WSP family are optimised for low-bandwidth bearer networks with relatively long latency.

5. WSP mapping to DM

The following sections define the requirements for the binding of DM Messages to WSP.

5.1 Multiple messages Per Package

The WAP protocol expects to receive a response to every request sent to the WAP gateway. If there are multiple messages in a DM package to be sent, the DM Server **MUST** send a response to each message although the message is not the final one.

The next message can only be sent when the WSP layer in the WAP protocol has received a response.

Each DM Message MUST be transferred as a SyncML MIME media type within the body of a WSP request or response. When there are multiple DM Messages per DM package, each message is transferred in a separate WSP request or response; depending on whether it is a DM request or response.

The recipient of a DM package can determine if there are more DM messages in the package by the absence of the Final element in the body of the last received DM message. When the recipient receives a DM Message with the Final element, it is the final message within that DM package.

5.2 MIME header type requirement

DM Clients and DM Servers MUST support this header with either the "application/vnd.syncml.dm+xml" or "application/vnd.syncml.dm+wbxml" media type values.

5.3 Connection Oriented Session

This section describes how DM Client residing on a WAP client would initiate a DM connection oriented session, exchange DM Messages with the DM Server, suspend and resume the session, and then finally close down the established session.

5.3.1 Session establishment, S-Connect

During a WAP session establishment, a WAP client connects to a WAP gateway. A part of this is the so-called capability negotiation, during which the server and client negotiate the features supported. Furthermore, attributes that are static throughout the sessions are exchanged (static headers).

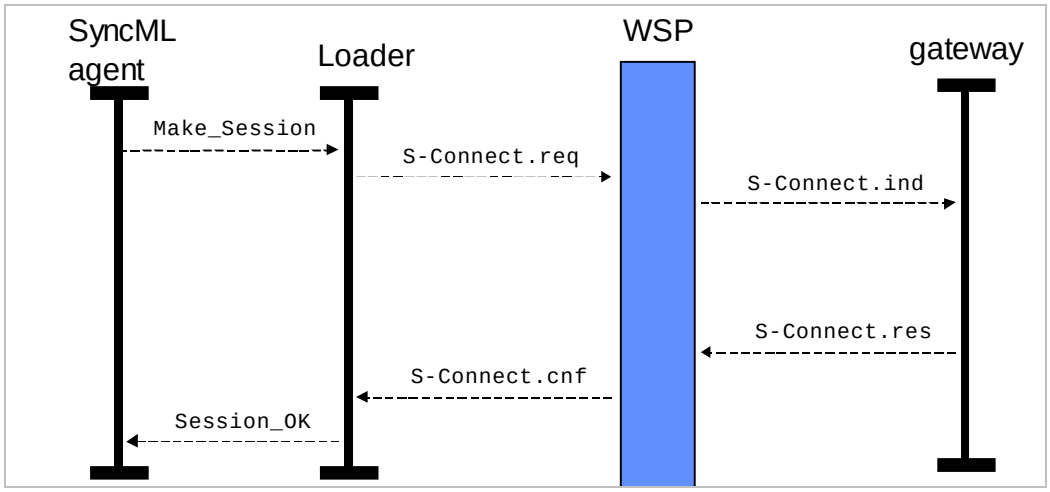

Figure 1: Session Establishment

In the example, the Loader implements an interface for the user agent to initiate a session, Make_Session. The Loader implements the HTTP protocol.

Seen from WSP, the session establishment starts by an S-Connect request to the WSP layer. The request looks as follows:

```
S-Connect.req(Server-Address , Client-Address, Client-Headers, Requested-
Capabilities)
```

In case of success, the connect confirmation returns from the WSP layer as follows:

```
S-Connect.cnf(Server-Headers , Negotiated-Capabilities)
```

5.3.2 Exchanging DM Messages

Once a session is established, the DM Client can start exchanging DM Messages with the DM Server using the S-MethodInvoke and S-MethodResult primitives.

WAP maps the HTTP 1.1 methods; i.e. requests will be done using standard HTTP 1.1 methods. The header and bodies of the HTTP methods are not used by the WAP stack, and they are passed transparently.

The following example shows a simple POST request from the client.

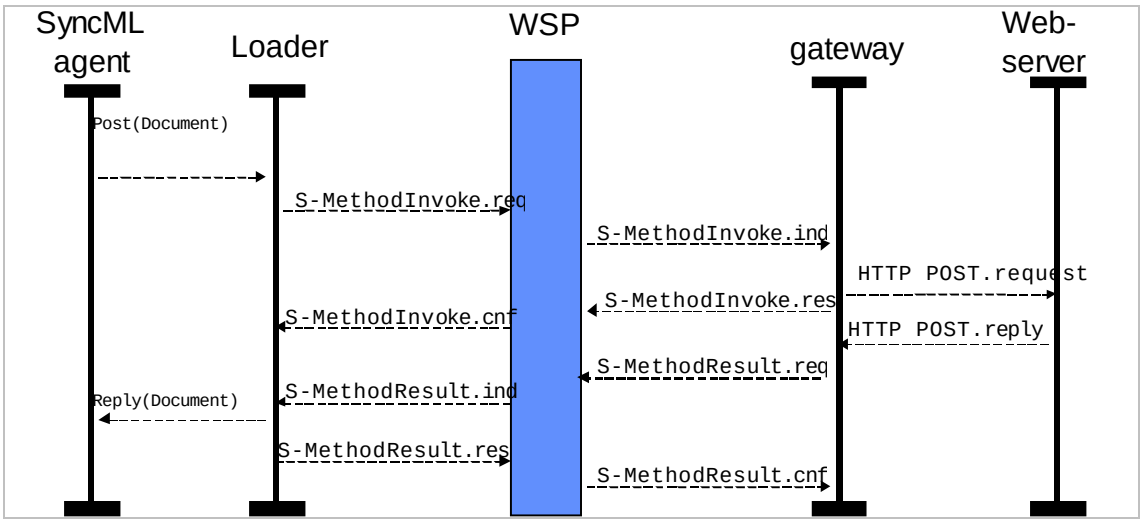

Figure 2: MethodInvoke using HTTP POST

In the implementation example depicted in Figure 2, the DM Client requests a DM Message to be posted to the DM Server using the interface made available by the Loader. In a response, the DM Server returns a response DM Message back to the client.

5.3.2.1 HTTP header requirement

The HTTP header information is passed transparently over the WAP protocol. But In order to enable the Web server to decode the posted information the same header information requirements apply for sending DM Messages over WSP as for sending DM Messages over HTTP as described in **Error! Reference source not found.**.

5.3.2.2 S-MethodInvoke

The syntax of the MethodInvoke is as follows:

```
S-MethodInvoke.req(ClientTransActionID, Method, RequestURI, RequestHeaders,
RequestBody)
```

The HTTP methods supported by WSP are GET, OPTIONS, HEAD, DELETE, TRACE, POST and PUT. Of all the HTTP methods supported by WSP, the DM functionality only requires the POST method. Once the gateway has processed the request (i.e. forwarded it to the web-server), a confirmation is sent back to the client through the WSP layer. The syntax of the S-MethodInvoke-confirmation is:

```
S-MethodInvoke.cnf(ClientTransactionID)
```

5.3.2.3 S-MethodResult

When the gateway receives the resource requested with the S-MethodInvoke primitive, it send a S-MethodResult request to the WSP layer of the client, which forwards the request to the user agent as a S-MethodResult-indication of the following format:

```
S-MethodResult.ind(ClientTransactionID, Status, ResponseHeaders, ResponseBody)
```

Once the indication is received, the client SHOULD reply to the WSP with a S-MethodResult response:

```
S-MethodResult.res(ClientTransactionID, Acknowledgement Headers)
```

5.3.3 Temporarily suspending the session, S-Suspend and S-Resume

WSP allows for the application layer to suspend a session. Suspending a session means that the sessions can no longer be used to communicate through until the session is resumed.

```
S-Suspend.req()
```

The indication coming back from WSP is of the following format:

```
S-Suspend.req(Reason)
```

5.3.4 Session close-down, S-Disconnect

The Disconnect primitive is used for terminating the active session.

5.4 Connectionless service

The connectionless service offered by WSP offers a connectionless, and potentially unreliable, data exchange service. Following example shows a POST request using the connectionless service.

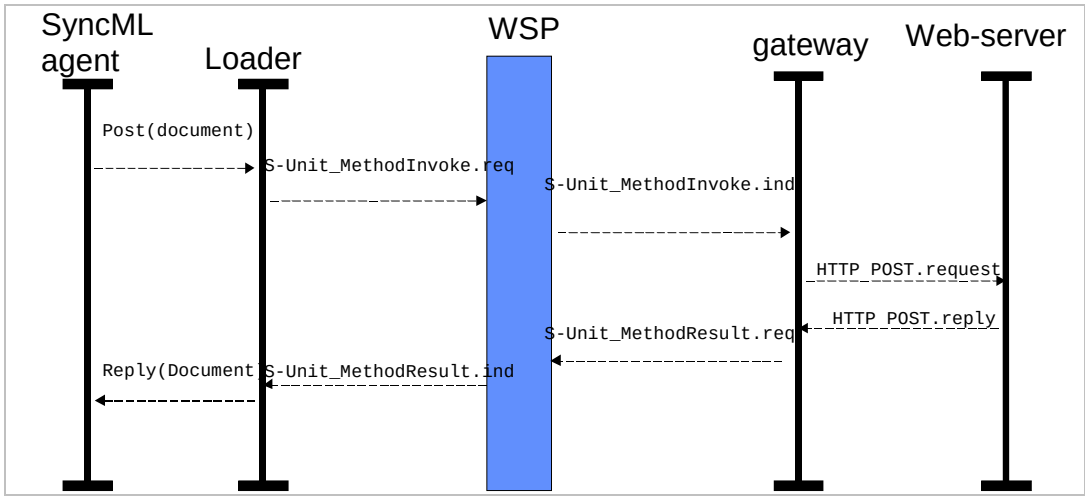

Figure 3: Connectionless Unit_MethodInvoke using HTTP POST

Only two primitives are supported by the connectionless service, MethodInvoke and Push. They both work as with the Session Oriented Service, but without the confirmation. Refer to the Session Oriented Service for details.

5.5 Pushing data from the server to the client

Pushing data from a server to a client, using Push OTA [PUSHOTA] (WSP or HTTP) can be used to push a DM Message to a DM Client (e.g. sending a DM Bootstrap or DM Sessionless message). This functionality is similar to the push of DM Notification [DMNOTI]. The model is as shown below.

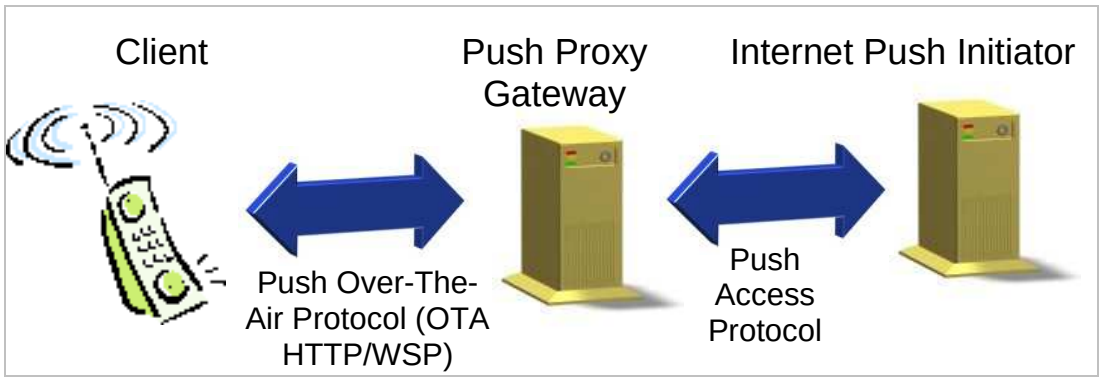

Figure 4: Push Model

The initiator of the push uses HTTP POST to send a XML document to the Push Proxy Gateway (PPG). The push message consists of two components; a control entity (containing e.g. information about when the push message expires) and the content to be pushed to the client. All WAP content types can be pushed.

When pushing DM Message from the DM Server to the DM Client, the following rules MUST be adhered to:

- The Content-Type header MUST include the MIME media type, *application/vnd.syncml.dm.+xml* or *application/vnd.syncml.dm+wbxml* for Device Management usage. The Content-Type binary code MUST be 0x42 (for wbxml) or 0x43 (for xml).
- The X-WAP-Application-ID header MUST include the application-id x-wap-application:syncml.dm for Device Management packages. The application-id binary code MUST be 0x07 for Device Management.

The PUSH model defined two different modes; confirmed and non-confirmed. The confirmed push requires an active WSP session. If no such session is running, a session establishment request can be issued from the PPG to a special application residing in the client. This application will initiate the session, where-after the push can be accomplished.

The non-confirmed push messages do not require a session.

A special push dispatcher in the client receives the push message, and forwards it to the right application, determined by the application identifier in the push message.

Once the right application receives the push content, it might choose to pull more content from a server. This is done using the standard WSP protocols as described in this document.

As such, the client that receives the pushed content doesn't interface directly to WSP.

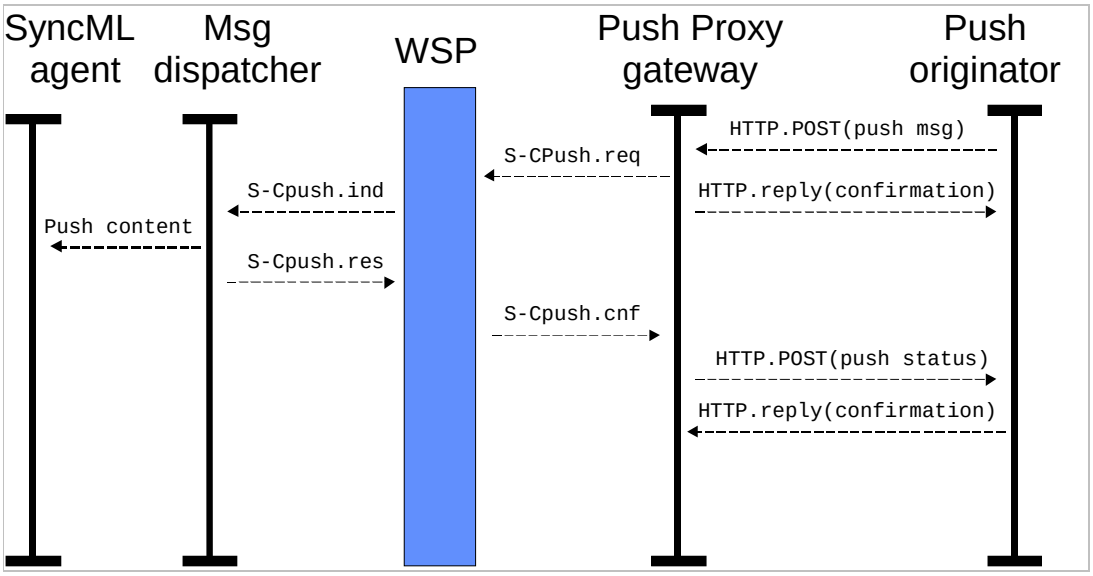

Figure 5: Push scenario

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
N/A	N/A	No prior version

A.2 Draft/Candidate Version 1.3 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-SyncML_WSPBinding-V1_3	27 Aug 2009	All	New baseline, based on OMA-TS-SyncML-WSPBinding-V1_2-20070221-A.
	25 Nov 2009	All	Incorporated CR: OMA-DM-DM13-2009-0098R02-CR_WSPBinding_cleanup
	10 Dec 2009	2.1, App A	Added 25 Nov 2009 row in history box Formatting clean-up
Draft Versions OMA-TS-DM_WSPBinding-V1_3	08 Jan 2010	All	General editorial clean-up by DSO
	03 May 2010	2.1, 5.5	Incorporated CR: OMA-DM-DM13-2010-0072R01-CR_WSP_MIME Sorting of references in alphabetical order
Candidate Version OMA-TS-DM_WSPBinding-V1_3	25 May 2010	N/A	Status changed to Candidate by TP Ref # OMA-TP-2010-0221-INP_DM_V1.3_ERP_and_ETR_for_Candidate_approval

Appendix B. Static Conformance Requirements (Normative)

The following specifies the static conformance requirements for DM over WSP devices conforming to this specification. The notation used in this appendix is specified in [IOPPROC].

B.1 Client Features

B.1.1 Client WSP Method Requirements

Item	Function	Ref.	Status	Requirement
DSDM-WSP-C-001	Support for POST method	5.3	M	
DSDM-WSP-C-002	Support for PUSH operation	5.5	O	

B.2 Server Features

B.2.1 Server WSP Method Requirements

Item	Function	Ref.	Status	Requirement
DSDM-WSP-S-001	Support for POST method	5.3	M	
DSDM-WSP-S-002	Support for PUSH operation	5.5	O	