Device Management HTTP Binding

Candidate Version 1.3 – 07 Dec 2010

Open Mobile Alliance

OMA-TS-DM_HTTPBinding-V1_3-20101207-C

Contents

.

1. Scope

This document describes the HTTP Binding for carrying DM Messages based on DM representation [DMPush].

2. References

2.1 Normative References

[DMPush] "OMA Device Management Push Binding", Open Mobile Alliance™, OMA-TS-DM_PushBinding-V1_3, URL:http://www.openmobilealliance.org/

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.1, Open Mobile Alliance™, OMA-IOP-Process-V1_1, 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

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998, URL:http://www.ietf.org/rfc/rfc2396.txt

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

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

[RFC2817] "Upgrading to TLS Within HTTP/1.1", R. Khare and S. Lawrence, May 2000, URL:http://www.ietf.org/rfc/rfc2817.txt

[RFC2818] "HTTP over TLS", E. Rescorla, May 2000, URL:http://www.ietf.org/rfc/rfc2818.txt

[RFC4279] " Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", P. Eronen, H. Tschofenig, December 2005, URL:http://www.ietf.org/rfc/rfc4279

[RFC4346] "The Transport Layer Security (TLS) Protocol Version 1.1", T. Dierks and E. Rescorla, April 2006, URL:http://www.ietf.org/rfc/rfc4346.txt

[SSL] "The SSL 3.0 Protocol", A. Frier, P. Karlton, and P. Kocher, Nov 1996

2.2 Informative References

[WSPBIND] "OMA Device Management WSP Binding Specification", Open Mobile Alliance™ , OMA-TS-SyncML_WSPBinding-V1_2, 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.

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

3.2 Definitions

Cache A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server can include a cache, though a cache cannot be used by a server that is acting as a tunnel.

Cacheable A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. The rules for determining the cacheability of HTTP responses are defined in section 13 of [RFC2616]. Even if a resource is cacheable, there can be additional constraints on whether a cache can use the cached copy for a particular request.

Connection A transport layer virtual circuit established between two programs for the purpose of communication.

Entity The information transferred as the payload of a HTTP request or HTTP response. An entity consists of meta-information in the form of entity-header fields and content in the form of an entity-body, as described in section 7 of [RFC2616].

HTTP Client A program that establishes connections for the purpose of sending HTTP requests.

NOTE: In this document, when the term "client" appears alone, it refers to a HTTP client, not a DM Client.

HTTP Gateway A HTTP server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client might not be aware that it is communicating with a gateway.

HTTP Message The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax defined in section 4 of [RFC2616] and transmitted via the connection.

HTTP Proxy An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is needed for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering. Except where either transparent or non-transparent behavior is explicitly stated, the HTTP proxy requirements apply to both types of proxies.

HTTP Request An HTTP request message, as defined in section 5 of [RFC2616].

HTTP Response An HTTP response message, as defined in section 6 of [RFC2616].

HTTP Server An application program that accepts connections in order to service HTTP requests by sending back responses. Any given program might be capable of being both an HTTP client and a server; our use of these terms refers only to the role being performed by the program for a particular connection,

	rather than to the program's capabilities in general. Likewise, any server might act as an HTTP origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request. NOTE: In this document, when the term "server" appears alone, it refers to a HTTP server, not a DM Server.
Inbound/Outbound	Inbound and outbound refer to the request and response paths for messages: "inbound" means "traveling toward the origin server", and "outbound" means "traveling toward the user agent."
Representation	An entity included with a response that is subject to content negotiation, as described in section 12 of [RFC2616]. There can exist multiple representations associated with a particular response status.
Resource	A network data object or service that can be identified by a URI, as defined in section 3.2 of [RFC2616]. Resources can be available in multiple representations (e.g. multiple languages, data formats, size, and resolutions) or vary in other ways.
User Agent	The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.
Origin Server	The HTTP server on which a given resource resides or is to be created.
Tunnel	An intermediary program which is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel might have been initiated by an HTTP request. The tunnel ceases to exist when both ends of the relayed connections are closed.

3.3 Abbreviations

HTTP	Hypertext Transfer Protocol [RFC2616]
OMA	Open Mobile Alliance
SSL	Secure Socket Layer [SSL]
TLS	Transport Layer Security [RFC4346]
URI	Uniform Resource Identifier [RFC2396]
URL	Uniform Resource Locator [RFC2396]
WAP	Wireless Application Protocol
WBXML	WAP Binary XML Content Format
XML	Extensible Markup Language

4. Introduction

This document defines the binding requirements for communicating DM Messages over the Hypertext Transfer Protocol (HTTP) as defined by [RFC2616]. HTTP is an application-level protocol for distributing information between two Internet based applications. HTTP can be an ideal protocol for building wide area, distributed systems, such as the collaborative framework of the World-Wide Web (Web) environment. HTTP is expected to be the primary wireline protocol used by DM Clients and servers to communicate with each other. In addition, as wireless networks increase in bandwidth (e.g., deployment of GPRS networks) and are adapted to support the IP protocol, HTTP will become the preferred application protocol for connecting wireless DM Clients and servers.

The HTTP protocol defines a request/response form of communication between two network computers supporting HTTP applications. Normally, the originator of the request is called the HTTP client. And normally, the recipient of the request is called the HTTP server.

There are emerging Internet standards for notification or "push" technologies that will allow HTTP servers to also originate HTTP requests. However, this technology is not currently included in this version of the specification.

HTTP is an ideal protocol for connecting a SyncML client to a DM Server; and vice a versa. Several MIME media types have been registered for DM. This makes it adapted for transfer over HTTP. HTTP is widely supported across the Internet. There are numerous examples of common implementations of both client and server HTTP applications. HTTP servers can be adapted to support new HTTP-based application in a straightforward manner. DM Messages can be transferred across the Internet and pass through "firewalls", at the perimeter of individual intranets with relative ease.

HTTP communication usually takes place over a TCP/IP connection. The default TCP port for HTTP is port 80, the default port for HTTPS / SSL / TLS is 443, but other unregistered ports can also be used. HTTP can also be implemented on top of any other reliable transport service.

An HTTP request message consists of start line containing a request method, target URI, and protocol version; followed by a MIME-like request header lines containing meta-information about the request; followed by a blank line; followed by a possible MIME entity body content. The server responds with a status line, including the message's protocol version and a response code; followed by MIME-like response header lines containing server information and entity meta-information; followed by a blank line; followed by an optional MIME entity body content.

5. HTTP Bindings

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

5.1 TCP Transport Service

HTTP communication usually takes place over a TCP connection. This binding does not require, but does assume such a connection. If the HTTP communication takes place over another transport service, then requirements similar to those defined here for TCP need to be followed. The following sections describe requirements for connection, disconnection and abort of the TCP connection.

5.1.1 Connection

Prior to an HTTP client connecting to an HTTP server, the DM Client makes a TCP connection using the TCP open operation between the client machine and the server machine. The client can use the Internet address resolution for the HTTP connection URL. The default port is 80. However, another port can be used. The choice of the appropriate port can be a requirement of the provisioning of the HTTP client to the HTTP server. After creating a successful TCP connection between the client and server machines, the HTTP connection between the client and server machines can be established by the DM Client. It cannot be assumed that the TCP connection is kept open between each HTTP request and response sequence (i.e., HTTP version 1.1 or higher protocol is being used). Even though HTTP version 1.1 presumes persistent connections, there can have occurred an anomaly such as a timeout condition or "denial of service" counter-measures on the origin server. Hence, the DM Client SHOULD be prepared for reconnection between HTTP requests.

5.1.2 Connection Options

The default port is 80. However, the provisioning of the HTTP client to the HTTP server can require use of another port.

Persistent connections are supported by this specification and presumed by HTTP version 1.1, but are not REQUIRED by implementations conforming to this specification.

5.1.3 Disconnection

The DM Client is responsible for terminating the connection with a TCP close operation, when the connection is no longer needed.

If a persistent connection exists between HTTP requests, the HTTP connection is closed by the HTTP client after receipt of the HTTP response corresponding to the last DM request in the synchronization session (i.e., the DM request in the last DM package containing a Final element type specified at the end of the SyncBody).

5.1.4 Abort

Sometimes abnormal conditions arise which requires an application to break the TCP connection. In these cases, the TCP reset operation is initiated to terminate the TCP connection.

5.1.5 Timeouts

In the case of a server timeout, the DM Client SHOULD establish a new HTTP session with the HTTP server and attempt to resend the current DM package, beginning with the first DM command for which the DM Client has not received an acknowledgement. In the event that the DM Client requested that no responses be sent, the DM Client SHOULD begin retransmitting with the first DM command in the DM package.

5.2 Exchanging DM Messages

Once an HTTP connection has been established, one or more DM Message are transferred over the connection, by the DM Client, in the entity body of HTTP requests or received from the server in the entity body of HTTP responses.

The POST method is used to transfer the DM Message in an HTTP request.

The body of the HTTP request MUST always include a DM Message. The content-type for the body MUST be the appropriate DM Message content type.

Generally, the Request-URI, in the start line, is the URI "path" component of the intended recipient resource for the request. The URI of the origin server is specified by the value in the Host request header.

The Request-URI can also be "*" to indicate that the request is intended for the origin server indicated by the absolute URI specified in the Host request header. In this case, the origin server would be the DM Server. However, in general, the DM Server will be a resource (e.g., servlet) under the origin server.

The HTTP response will always include the transport response status code. The body of the HTTP Response MUST always include a DM Message. The content-type for the body SHOULD be either the "application/vnd.syncml.dm+xml" or "application/vnd.syncml.dm+wbxml".

5.2.1 Single Message Per Package

The following is a snippet of the minimal HTTP start line and request headers for a simple HTTP request where the body of the request is the clear-text, DM Message media type.

```
POST ./servlet/dm HTTP/1.1
Host: www.devicemgmt.org
Content-Type: application/vnd.syncml.dm+xml; charset="utf-8"
Content-Length: 1023
Accept: application/vnd.syncml.dm+xml
```

Were the body, the binary, WBXML **Error! Reference source not found.** DM Message media type, then no content encoding is necessary. HTTP does not support the Content-Transfer-Encoding of MIME, in any case.

The following is a snippet of the minimal HTTP for an example for a simple HTTP response where the body in the response is the clear-text, DM Message media type, as specified in the Accept request header in the HTTP request.

```
HTTP/1.1 200 OK
Content-Type: application/vnd.syncml.dm+xml; charset="utf-8"
Content-Length: 1023

-- HTTP body, represented in a format consistent with the DM Message
type follows --
```

5.2.2 Multiple Messages Per DM Package

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

This version of the specification does NOT support transferring DM Messages across HTTP using a "multipart” MIME media type.

5.3 Transport Commands

HTTP uses a number of types of commands. The following sections specify static conformance requirements for use of these commands in the DM HTTP binding.

5.3.1 Methods

The following table summarizes the support for the HTTP methods in the DM HTTP binding.

Method	Client Requirements	Server Requirements
OPTIONS	MAY	MAY
GET	MAY	MAY
HEAD	MAY	MAY
POST	MUST	MUST
PUT	MAY	MAY
DELETE	MAY	MAY
TRACE	MAY	MAY
CONNECT	MAY	MAY

The DM Client MUST use the POST or MAY use the CONNECT method (if supported) to send DM requests to the DM Server. The CONNECT method is used to initiate an SSL [SSL] / TLS [RFC4346] session to authenticate the HTTP client to the HTTP server. A typical HTTP request start line would look like the following:

```
POST ./servlet/dm HTTP/1.1
Host: http://www.dm.host.com
```

In this example, the HTTP client is specifying a "./servlet/dm" path component for the Request-URI, which is a particular DM resource on the HTTP server. The "Host" HTTP header field is needed to specify the absolute URI for the HTTP server.

The other HTTP methods MAY be supported by implementations conforming to this specification. However, they are not used at this time by the DM Client.

5.3.2 General Headers

The following table summarizes the support for the HTTP general headers in the DM HTTP binding.

General Header	Client Requirements	Server Requirements
Cache-Control	MUST	MUST
Connection	MAY	MAY
Content-Encoding	MAY	MAY
Date	MAY	MAY
Pragma	MAY	MAY
Trailer	MAY	MAY
Transfer-Encoding	MUST	MUST
Upgrade	MAY	MAY
Via	MAY	MAY
Warning	MAY	MAY

The Cache-Control general header is used to force control over the caching mechanisms in the request/response chain between the HTTP client and the HTTP server. Implementations conforming to this specification MUST support this header. Use of this header with the no-store parameter will assure that SyncML messages sent by the DM Client and DM Server are

not stored in cache storage. This will greatly assure that DM Server and DM Client are processing the messages sent by the DM Clients and DM Servers, respectively. The following is an example of the typical specification for this header:

```
Cache-Control: no-store
```

In addition, implementations conforming to this specification MUST support the private Cache-Control option to assure that responses do not get cached and possibly used by agents other than the DM Client agent or the DM Server agent.

```
Cache-Control: private
```

The Content-Encoding header is used as a modifier to the media-type. When present, its value indicates what additional content coding have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a document to be compressed without losing the identity of its underlying media type. Implementations conforming to this specification MAY support Content-Encodings other than identity. Note that since DM messages are frequently used in bandwidth constrained environments, the use of compression may be especially desirable. The following is an example of how this header is specified to indicate that gzip compression was used on the message.

```
Content-Encoding: gzip
```

The Transfer-Encoding general header is used to indicate what (if any) type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient. Implementations conforming to this specification MUST support the chunked Transfer-Encoding option.

```
Transfer-Encoding: chunked
```

The other general headers MAY be supported by implementations conforming to this specification.

5.3.3 Request Headers

The following table summarizes the support for the HTTP request headers in the DM HTTP binding.

Request Header	Client Requirements	Server Requirements
Accept	MUST	MUST
Accept-Charset	MUST	MUST
Accept-Encoding	MAY	MAY
Accept-Language	MAY	MAY
Authorization	MAY	MAY
Expect	MAY	MAY
From	MAY	MAY
Host	MAY	MAY
If-Match	MAY	MAY
If-Modified-Since	MAY	MAY
If-None-Match	MAY	MAY
If-Range	MAY	MAY
If-Unmodified-Since	MAY	MAY
Max-Forwards	MAY	MAY
Proxy-Authorization	MAY	MAY
Range	MAY	MAY
Referer	MAY	MAY
TE	MAY	MAY
User-Agent	MUST	MUST

The Accept request header is used to specify which media types are acceptable in the response.

The following is an example of how this header is specified to indicate that the client expects responses formatted according to the binary, WBXML **Error! Reference source not found.** representation of device management

```
Accept: application/vnd.syncml.dm+wbxml
```

The Accept-Charset request header is used to specify which character sets are acceptable in the response. DM Servers MUST support this header with the "UTF-8" character set value. DM Clients SHOULD support the "UTF-8" character set. DM Clients MAY support additional, IANA registered character set values. DM Client implementations not supporting UTF-8 SHOULD take careful consideration of the potential impact of lack of UTF-8 support on interoperability of the device. If a recipient is unable to provide support for the character set encoding specified in the Accept-Charset request headers sent by the originator, the recipient MUST send to the originator a HTTP status 406, "Not acceptable". This is in keeping with [RFC2616]. Note that there will be no DM Message sent with this response. The following is an example of how this header is specified to indicate that the client expects responses formatted into the UTF-8 character set:

```
Accept-Charset: UTF-8
```

The Authorization request header is used by an HTTP client to authenticate itself to the HTTP server. DM Clients and DM Servers MAY support this header with the authorization values for "Digest Access Authentication" authentication schemes, as specified in [RFC2617]. The following is an example of how this header is specified to allow the HTTP client to authenticate itself with a Base64 character encoding of a userid of Aladdin and password of open sesame.

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

The Accept-Encoding request header is similar to Accept, but restricts the content-codings (Section 3.5 of [RFC2616]) that are acceptable in the response. DM Clients and DM Servers MAY support this header. Note that since DM messages are frequently used in bandwidth constrained environments, the use of compression may be especially desirable. The following is an example of how this header is specified to allow the HTTP client to request particular compression types.

```
Accept-Encoding: gzip,deflate
```

The Proxy-Authorization request header is similar to the Authorization header except that it is specified by the HTTP client and used only by the next proxy in the connection chain. DM Clients and DM Servers MAY support this header and with the authorization values for "Digest Access Authentication" authentication schemes, as specified in [RFC2617]. The following is an example of how this header is specified to allow the HTTP client to authenticate itself with a userid of Aladdin and password of open sesame.

```
Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

The User-Agent request header identifies the type of user agent originating the request. This information is useful for the HTTP server to provide automated recognition of user agents for the sake of tailoring responses to avoid particular limitations or to take advantage of special features in the HTTP client. Implementations conforming to this specification MUST support this header and provide it in all HTTP requests. The following is an example of the usage of this header.

```
User-Agent: Foo Bar DM Products v3.4
```

The other request headers MAY be supported by implementations conforming to this specification.

5.3.4 Response Headers

The following table summarizes the support for the HTTP response headers in the DM HTTP binding.

Method	Client Requirements	Server Requirements
Accept-Ranges	MAY	MAY
Age	MAY	MAY
Allow	MAY	MAY
Authentication-Info	MAY	MAY

Etag	MAY	MAY
Location	MAY	MAY
Proxy-Authenticate	MAY	MAY
Retry-After	MAY	MAY
Server	MAY	MAY
Vary	MAY	MAY
WWW-Authenticate	MAY	MAY

The Authentication-Info response header is defined by [RFC2617]. The header is used by an HTTP proxy or server to provide information back to the HTTP client about a successful HTTP client authentication. Implementations conforming to this specification MAY support this header with the Authentication-Information directives "nextnonce" and "rspauth". The former directive is used to specify the nonce to be used by the client for a future authentication. The nonce value SHOULD be a Base64 formatted string. The nextnonce value string MUST be 64 octets or less in length. The latter directive is used by the HTTP proxy or server to authenticate itself to the HTTP client. The following example shows how an HTTP server might use this response header to provide authentication credentials to an HTTP client that has successfully authenticated itself with the HTTP server.

```
Authentication-Info: nextnonce="Bruce"
rspauth="edd30630e82fabdc1e895d1d3a4c0450"
```

The Proxy-Authenticate response header is used by an HTTP proxy to challenge the authority of the HTTP client issuing it an HTTP request. Implementations conforming to this specification MAY support this header with the challenge values for "Basic" and "Digest Access Authentication" authentication schemes and the "Realm", "Domain", "Nonce", "Stale" and "Algorithm" authentication parameters, as specified in [RFC2617]. The nonce value SHOULD be a Base64 formatted string. The "MD5" algorithm MAY be supported by implementations that conform to this specification. Other algorithms can also be supported. The following is an example of this header being used by an HTTP proxy to challenge a HTTP client with the Digest authentication scheme for the http://www.dm.host.com realm.

```
Proxy-Authenticate: Digest
Domain="http://www.devicemgmt.org/servlet/dm"
```

The WWW-Authenticate response header is used by the HTTP server to challenge the authority of the HTTP client issuing it an HTTP request. Implementations conforming to this specification MAY support this header with the challenge values for "Basic" and "Digest Access Authentication" authentication schemes and the "Realm", "Domain", "Nonce", "Stale" and "Algorithm" authentication parameters, as specified in [RFC2617]. The nonce value SHOULD be a Base64 formatted string. The "MD5" algorithm MAY be supported by implementations that conform to this specification. Other algorithms can also be supported. The following is an example of this header being used by an HTTP server to challenge an HTTP client with the Basic authentication scheme for the WallyWord@dm.host.com realm.

```
WWW-Authenticate: Basic Realm="WallyWorld@dm.host.com"
```

The other response headers MAY be supported by implementations conforming to this specification.

5.3.5 Pushing Data from the server to the client

See the Push Binding [DMPush] for information on how to push a DM Message to the DM Client.

5.4 Security

HTTP client and HTTP server authentication is based on the mechanism defined in [RFC2617]. Implementations conforming to this implementation MUST support this mechanism for "Basic" and "Digest Access Authentication". The Base64 character encoded "Basic" and "MD5" algorithm of the "Digest Access Authentication" authentication schemes MAY be supported.

The HTTP headers and parameters that MUST be supported are described in the previous sections for request and response headers.

The Content-MD5 header field can be used to provide a message integrity check of the DM Message in the body. Use of this header is a good idea for detecting accidental modification of the entity-body in transit, but is not proof against malicious attack.

HTTP proxy and HTTP server implementations conforming to this specification MAY support both the ability to challenge unauthenticated requests and also accept authentication request headers in a request; which will not require subsequent challenge responses unless some part of the credential is incorrect. The latter requirement is REQUIRED to address the need for minimal request/response traffic for mobile networks.

The authentication mechanisms defined by [RFC2617] address protecting the authentication credentials. However, the remainder of the HTTP request and response messages are available to the eavesdropper. For more robust security for the HTTP connection, SSL [SSL], TLS 1.1 [RFC4346], TLS 1.2 [RFC5246], PSK-TLS [RFC4279], HTTPS, or some form of upgrading to TLS over HTTP [RFC2817] [RFC2818] SHOULD be used.

When operating over HTTP:

- The Server and Client MUST support TLS 1.1.
- The Server and Client MUST use TLS 1.0 or SSL3.0 or TLS 1.1 or PSK-TLS or TLS 1.2.
- The Server SHOULD support TLS 1.0 or TLS 1.2 [RFC5246], SSL 3.0 [SSL3.0] and PSK-TLS [RFC4279].
- The Client MUST identify that the Server is using TLS1.0 or SSL3.0 or TLS 1.1 or PSK-TLS or TLS 1.2.
- A Session SHALL NOT take place over SSL2.0 or less.
- The Server MUST support both of the following cipher suites, both of which provide authentication, confidentiality and integrity, when using an SSL3.0 session
    - SSL_RSA_WITH_RC4_128_SHA
    - SSL_RSA_WITH_3DES_EDE_CBC_SHA
- The Client MUST support at least one of the following cipher suites, both of which provide authentication, confidentiality and integrity, when using an SSL3.0 session
    - SSL_RSA_WITH_RC4_128_SHA
    - SSL_RSA_WITH_3DES_EDE_CBC_SHA
- The Server MUST support both of the following cipher suites, both of which provide authentication, confidentiality and integrity, when using an PSK-TLS session
    - TLS_PSK_WITH_AES_128_GCM_SHA256
    - TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
    - TLS_RSA_PSK_WITH_AES_128_GCM_SHA256
    - TLS_PSK_WITH_AES_128_CBC_SHA256
    - TLS_DHE_PSK_WITH_AES_128_CBC_SHA256
    - TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
- The Client MUST support at least one of the following cipher suites, both of which provide authentication, confidentiality and integrity, when using an PSK-TLS session

  - TLS_PSK_WITH_AES_128_GCM_SHA256
  - TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
  - TLS_RSA_PSK_WITH_AES_128_GCM_SHA256
  - TLS_PSK_WITH_AES_128_CBC_SHA256
  - TLS_DHE_PSK_WITH_AES_128_CBC_SHA256
  - TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
- The Server MUST only accept the usage of cipher suites with all the following characteristics:
  - At least 128 bit symmetric keys;
  - Provide key exchange;
  - Provide signing;
  - Provide bulk encryption;
  - Provide message authentication.
- The Server MUST support the requirements relating to certificates and certificate processing in section 6.3 and 6.4 of the WAP TLS Profile and Tunneling, [WAP-219-TLS].
- If the Client supports TLS1.0, it MUST support the requirements relating to certificates and certificate processing in section 6.3 and 6.4 of the WAP TLS Profile and Tunneling, [WAP-219-TLS].

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
N/A	N/A	No prior 1.3 version

A.2 Draft/Candidate Version 1.3 History

Document Identifier	Date	Sections	Description
Draft versions OMA-TS-SyncML_HTTPBinding-V1_3	30 Jul 2009	2.1 5.4 B.1.5	Based on OMA-TS-SyncML_HTTPBinding-V1_2_1 Incorporated CRs: OMA-DM-DM13-2009-0028R03-CR_HTTP_Security OMA-DM-DM13-2009-0050-CR_HTTP_security
	08 Sep 2009	5.4	Incorporated CRs OMA-DM-DM13-2009-0057R01-CR_Cypher_Suitability OMA-DM-DM13-2009-0063R02-CR_TLS12.
	24 Nov 2009	All	Incorporated CRs: OMA-DM-DM13-2009-0073R01-CR_BugFix_CipherSuite OMA-DM-DM13-2009-0096R02-CR_HTTPBinding_Cleanup
Draft versions OMA-TS-DM_HTTPBinding-V1_3	08 Jan 2010	All	Editorial clean-upby DSO and renaming of the TS from “OMA-TS-SyncML_HTTPBinding-V1_3” to “OMA-TS-DM_HTTPBinding-V1_3” as agreed by DM on 07 Jan 2010
	03 Feb 2010	5.2.1	Incorporated CR: OMA-DM-DM13-2010-0006-CR_HTTPBinding_Content_Type_Fix.doc
	10 Feb 2010	All	Light editorial clean-up by DSO
	03 May 2010	5.3.5	Incorporated CR: OMA-DM-DM13-2010-0070R02-CR_HTTP_MIME
	05 May 2010	2.1	Modified reference [PushOTA] to harmonize with [PushOTA] reference in other permanent documents
Candidate Version OMA-TS-DM_HTTPBinding-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
Draft versions OMA-TS-DM_HTTPBinding-V1_3	12 Aug 2010	2.1, 5.3.5, 5.4	Incorporated CRs: OMA-DM-DM13-2010-0089R01-CR_Remove_HTTP_Push OMA-DM-DM13-2010-0102R01-CR_Mandate_TLS1.1_interop_problem
	25 Aug 2010	App B	Incorporated CRs: OMA-DM-DM13-2010-0092R02-CR_HttpBinding_SCR_correction_and_Missing_Info
Candidate Version OMA-TS-DM_HTTPBinding-V1_3	07 Dec 2010	N/A	Status changed to Candidate by TP Ref #OMA-TP-2010-0502-INP_DM_V1_3_ERP_and_ETR_for_Candidate_re_approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 Client Features

B.1.1 Client HTTP Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-001	Support for HTTP Client	4	O	DSDM-HTTP-C-005 AND DSDM-HTTP-C-010 AND DSDM-HTTP-C-015 AND DSDM-HTTP-C-019 AND DSDM-HTTP-C-020 AND DSDM-HTTP-C-037

B.1.2 Client HTTP Method Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-002	Support for OPTIONS Method	5.3.1	O	
DSDM-HTTP-C-003	Support for GET Method	5.3.1	O	
DSDM-HTTP-C-004	Support for HEAD Method	5.3.1	O	
DSDM-HTTP-C-005	Support for POST Method	5.3.1	O	
DSDM-HTTP-C-006	Support for PUT Method	5.3.1	O	
DSDM-HTTP-C-007	Support for DELETE Method	5.3.1	O	
DSDM-HTTP-C-008	Support for TRACE Method	5.3.1	O	
DSDM-HTTP-C-009	Support for CONNECT Method	5.3.1	O	

B.1.3 Client HTTP General Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-010	Support for Cache-Control General Header	5.3.2	O	
DSDM-HTTP-C-011	Support for Connection General Header	5.3.2	O	
DSDM-HTTP-C-049	Support for Content-Encoding Header	5.3.2	O	
DSDM-HTTP-C-012	Support for Date General Header	5.3.2	O	
DSDM-HTTP-C-013	Support for Pragma General Header	5.3.2	O	
DSDM-HTTP-C-014	Support for Trailer General Header	5.3.2	O	
DSDM-HTTP-C-015	Support for Transfer-Encoding General Header	5.3.2	O	
DSDM-HTTP-C-016	Support for Upgrade General Header	5.3.2	O	
DSDM-HTTP-C-017	Support for Via General Header	5.3.2	O	
DSDM-HTTP-C-018	Support for Warning General Header	5.3.2	O	

B.1.4 Client HTTP Request Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-019	Support for Accept Request Header	5.3.3	O	
DSDM-HTTP-C-020	Support for Accept-Charset Request Header	5.3.3	O	
DSDM-HTTP-C-021	Support for Accept-Encoding Request Header	5.3.3	O	
DSDM-HTTP-C-022	Support for Accept-Language Request Header	5.3.3	O	
DSDM-HTTP-C-023	Support for Authorization Request Header	5.3.3	O	
DSDM-HTTP-C-024	Support for Expect Request Header	5.3.3	O	
DSDM-HTTP-C-025	Support for From Request Header	5.3.3	O	
DSDM-HTTP-C-026	Support for Host Request Header	5.3.3	O	
DSDM-HTTP-C-027	Support for If-Match Request Header	5.3.3	O	
DSDM-HTTP-C-028	Support for If-Modified-Since Request Header	5.3.3	O	
DSDM-HTTP-C-029	Support for If-None-Match Request Header	5.3.3	O	
DSDM-HTTP-C-030	Support for If-Range Request Header	5.3.3	O	
DSDM-HTTP-C-031	Support for If-Unmodified-Since Request Header	5.3.3	O	
DSDM-HTTP-C-032	Support for Max-Forwards Request Header	5.3.3	O	
DSDM-HTTP-C-033	Support for Proxy-Authorization Request Header	5.3.3	O	
DSDM-HTTP-C-034	Support for Range Request Header	5.3.3	O	
DSDM-HTTP-C-035	Support for Referer Request Header	5.3.3	O	
DSDM-HTTP-C-036	Support for TE Request Header	5.3.3	O	
DSDM-HTTP-C-037	Support for User-Agent Request Header	5.3.3	O	

B.1.5 Client HTTP Response Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-038	Support for Accept-Ranges Response Header	5.3.4	O	
DSDM-HTTP-C-039	Support for Age Response Header	5.3.4	O	
DSDM-HTTP-C-040	Support for Allow Response Header	5.3.4	O	
DSDM-HTTP-C-041	Support for Authentication-Info Response Header	5.3.4	O	
DSDM-HTTP-C-042	Support for Etag Response Header	5.3.4	O	
DSDM-HTTP-C-043	Support for Location Response Header	5.3.4	O	
DSDM-HTTP-C-044	Support for Proxy-Authenticate Response Header	5.3.4	O	

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-045	Support for Retry-After Response Header	5.3.4	O	
DSDM-HTTP-C-046	Support for Server Response Header	5.3.4	O	
DSDM-HTTP-C-047	Support for Vary Response Header	5.3.4	O	
DSDM-HTTP-C-048	Support for WWW-Authenticate Response Header	5.3.4	O	

B.1.6 Client HTTP Security Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-C-049	Support HTTP	Section 5.4	O	DSDM-HTTP-C-050 AND DSDM-HTTP-C-051 AND DSDM-HTTP-C-054
DSDM-HTTP-C-050	Identifying that the Server is using TLS1.0 or TLS1.1 or SSL3.0 or PSK-TLS or TLS1.2	Section 5.4	O	
DSDM-HTTP-C-051	Use at least one of the security mechanisms of SSL3.0, TLS1.0, TLS 1.1, TLS1.2 or PSK-TLS	Section 5.4	O	
DSDM-HTTP-C-052	Support for at least one cipher suite, when using SSL3.0: SSL_RSA_WITH_RC4_128_SHA and SSL_RSA_WITH_3DES_EDE_CBC_SHA	Section 5.4	O	
DSDM-HTTP-C-053	Support for at least one cipher suite, when using PSK-TLS: TLS_PSK_WITH_AES_128_GCM_SHA256 , TLS_DHE_PSK_WITH_AES_128_GCM_SHA256, TLS_RSA_PSK_WITH_AES_128_GCM_SHA256, TLS_PSK_WITH_AES_128_CBC_SHA256, TLS_DHE_PSK_WITH_AES_128_CBC_SHA256, TLS_RSA_PSK_WITH_AES_128_CBC_SHA256	Section 5.4	O	
DSDM-HTTP-C-054	Support for TLS1.1	5.4	O	

B.2 Server Features

B.2.1 Server HTTP Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-001	Support for HTTP Server	4	O	DSDM-HTTP-S-005 AND DSDM-HTTP-S-010 AND DSDM-HTTP-S-015 AND DSDM-HTTP-S-019 AND

Item	Function	Ref.	Status	Requirement
				DSDM-HTTP-S-020 AND DSDM-HTTP-S-037

B.2.2 Server HTTP Method Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-002	Support for OPTIONS Method	5.3.1	O	
DSDM-HTTP-S-003	Support for GET Method	5.3.1	O	
DSDM-HTTP-S-004	Support for HEAD Method	5.3.1	O	
DSDM-HTTP-S-005	Support for POST Method	5.3.1	O	
DSDM-HTTP-S-006	Support for PUT Method	5.3.1	O	
DSDM-HTTP-S-007	Support for DELETE Method	5.3.1	O	
DSDM-HTTP-S-008	Support for TRACE Method	5.3.1	O	
DSDM-HTTP-S-009	Support for CONNECT Method	5.3.1	O	

B.2.3 Server HTTP General Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-010	Support for Cache-Control General Header	5.3.2	O	
DSDM-HTTP-S-011	Support for Connection General Header	5.3.2	O	
DSDM-HTTP-S-049	Support for Content-Encoding Header	5.3.2	O	
DSDM-HTTP-S-012	Support for Date General Header	5.3.2	O	
DSDM-HTTP-S-013	Support for Pragma General Header	5.3.2	O	
DSDM-HTTP-S-014	Support for Trailer General Header	5.3.2	O	
DSDM-HTTP-S-015	Support for Transfer-Encoding General Header	5.3.2	O	
DSDM-HTTP-S-016	Support for Upgrade General Header	5.3.2	O	
DSDM-HTTP-S-017	Support for Via General Header	5.3.2	O	
DSDM-HTTP-S-018	Support for Warning General Header	5.3.2	O	

B.2.4 Server HTTP Request Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-019	Support for Accept Request Header	5.3.3	O	
DSDM-HTTP-S-020	Support for Accept-Charset Request Header	5.3.3	O	
DSDM-HTTP-S-021	Support for Accept-Encoding Request Header	5.3.3	O	
DSDM-HTTP-S-022	Support for Accept-Language Request Header	5.3.3	O	
DSDM-HTTP-S-023	Support for Authorization Request Header	5.3.3	O	
DSDM-HTTP-S-024	Support for Expect Request Header	5.3.3	O	

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-025	Support for From Request Header	5.3.3	O	
DSDM-HTTP-S-026	Support for Host Request Header	5.3.3	O	
DSDM-HTTP-S-027	Support for If-Match Request Header	5.3.3	O	
DSDM-HTTP-S-028	Support for If-Modified-Since Request Header	5.3.3	O	
DSDM-HTTP-S-029	Support for If-None-Match Request Header	5.3.3	O	
DSDM-HTTP-S-030	Support for If-Range Request Header	5.3.3	O	
DSDM-HTTP-S-031	Support for If-Unmodified-Since Request Header	5.3.3	O	
DSDM-HTTP-S-032	Support for Max-Forwards Request Header	5.3.3	O	
DSDM-HTTP-S-033	Support for Proxy-Authorization Request Header	5.3.3	O	
DSDM-HTTP-S-034	Support for Range Request Header	5.3.3	O	
DSDM-HTTP-S-035	Support for Referer Request Header	5.3.3	O	
DSDM-HTTP-S-036	Support for TE Request Header	5.3.3	O	
DSDM-HTTP-S-037	Support for User-Agent Request Header	5.3.3	O	

B.2.5 Server HTTP Response Header Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-038	Support for Accept-Ranges Response Header	5.3.4	O	
DSDM-HTTP-S-039	Support for Age Response Header	5.3.4	O	
DSDM-HTTP-S-040	Support for Allow Response Header	5.3.4	O	
DSDM-HTTP-S-041	Support for Authentication-Info Response Header	5.3.4	O	
DSDM-HTTP-S-042	Support for Etag Response Header	5.3.4	O	
DSDM-HTTP-S-043	Support for Location Response Header	5.3.4	O	
DSDM-HTTP-S-044	Support for Proxy-Authenticate Response Header	5.3.4	O	
DSDM-HTTP-S-045	Support for Retry-After Response Header	5.3.4	O	
DSDM-HTTP-S-046	Support for Server Response Header	5.3.4	O	
DSDM-HTTP-S-047	Support for Vary Response Header	5.3.4	O	
DSDM-HTTP-S-048	Support for WWW-Authenticate Response Header	5.3.4	O	

B.2.6 Server HTTP Security Requirements

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-049	Supports HTTP	Section 5.4	O	DSDM-HTTP-S-051 AND DSDM-HTTP-S-055
DSDM-HTTP-S-050	Support the security mechanisms of SSL3.0, TLS1.0, TLS 1.1, TLS1.2 and PSK-TLS	Section 5.4	O	

Item	Function	Ref.	Status	Requirement
DSDM-HTTP-S-051	Use at least one of the security mechanisms of SSL3.0, TLS1.0, TLS 1.1, TLS1.2 or PSK-TLS	Section 5.4	O	
DSDM-HTTP-S-052	Support for TLS1.0	Section 5.4	O	DSDM-HTTP-S-051
DSDM-HTTP-S-053	Support for SSL 3.0	Section 5.4	O	DSDM-HTTP-S-054
DSDM-HTTP-S-054	Supporting these cipher suites: SSL_RSA_WITH_RC4_128_SHA and SSL_RSA_WITH_3DES_EDE_CBC_SHA	Section 5.4	O	
DSDM-HTTP-S-055	Support for TLS1.1	Section 5.4	O	
DSDM-HTTP-S-056	Support for PSK-TLS	Section 5.4	O	DSDM-HTTP-S-057
DSDM-HTTP-S-057	Supporting these cipher suites: TLS_PSK_WITH_AES_128_GCM_SHA256 , TLS_DHE_PSK_WITH_AES_128_GCM_SHA256, TLS_RSA_PSK_WITH_AES_128_GCM_SHA256, TLS_PSK_WITH_AES_128_CBC_SHA256, TLS_DHE_PSK_WITH_AES_128_CBC_SHA256, TLS_RSA_PSK_WITH_AES_128_CBC_SHA256	Section 5.4	O	
DSDM-HTTP-S-058	Support for TLS1.2	Section 5.4	O	