OMA Device Management Security

Approved Version 1.2 – 09 Feb 2007

Open Mobile Alliance

OMA-TS-DM_Security-V1_2-20070209-A

Contents

1. Scope

This document describes OMA-DM security requirements in general, and provides description of transport layer security, application layer security, etc. It also describes security mechanisms that are used to provide for integrity, confidentiality and authentication.

2. References

2.1 Normative References

[C.S0023-B_v1.0] “Removable User Identity Module For Spread Spectrum Systems”, 3GPP2 C.S0023-B version 1.0, URL:http://www.3gpp2.org/Public_html/specs/C.S0023-B_v1.0_040426.pdf

[DMBOOT] “OMA Device Management Bootstrap, Version 1.2”. Open Mobile Alliance™. OMA-TS-DM_Bootstrap-V1_2. URL:http://www.openmobilealliance.org

[DMNOTI] “OMA Device Management Notification Initiated Session, Version 1.2”. Open Mobile Alliance™. OMA-TS-DM_Notification-V1_2. URL:http://www.openmobilealliance.org

[DMPRO] “OMA Device Management Protocol, Version 1.2”. Open Mobile Alliance™. OMA-DM_Protocol-V1_2. URL:http://www.openmobilealliance.org

[DMTND] “OMA Device Management Tree and Description, Version 1.2”. Open Mobile Alliance™. OMA-TS-DM_TND-V1_2**.** URL:http://www.openmobilealliance.org

[GSM11.11] “Digital cellular Telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface”, (ETSI TS 100 977). URL:http://www.etsi.org/

[IOPPROC] “OMA Interoperability Policy and Process”, Version 1.1, Open Mobile Alliance™, OMA-IOP-Process-V1_1, URL:http://www.openmobilealliance.org/

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

[PROVSC] “Provisioning Smart Card Specification Version 1.1”. Open Mobile Alliance™. OMA-WAP-ProvSC-v1_1. URL:http://www.openmobilealliance.org

[RFC1321] “The MD5 Message-Digest Algorithm”. Network Working Group. April 1992. URL:http://www.ietf.org/rfc/rfc1321.txt

[RFC1521] “MIME (Multipurpose Internet Mail Extensions) Part One”. Network Working Group. September 1993. URL:http://www.ietf.org/rfc/rfc1521.txt

[RFC2104] “HMAC: Keyed-Hashing for Message Authentication”. Network Working Group. February 1997. URL:http://www.ietf.org/rfc/rfc2104.txt

[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”. Network Working group. June 1999. URL:http://www.ietf.org/rfc/rfc2616.txt

[SHA] “Secure Hash Standard”, NIST FIPS PUB 180-1, National Institute of Standards and Technology, U.S. Department of Commerce, DRAFT, May 1994. URL: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[SSL3.0] The SSL Protocol, Version 3.0, <draft-freier-ssl-version3-02.txt>, Transport Layer Security Working Group, Alan O. Freier et al, November 18, 1996. URL:http://www.netscape.com/eng/ssl3/draft302.txt

[TLS] RFC 2246 - The TLS Protocol Version 1.0, Network Working Group, January 1999. URL:http://www.ietf.org/rfc/rfc2246.txt

[TS102.221] “Smart Cards; UICC-Terminal interface; Physical and logical characteristics”, (ETSI TS 102 221), URL:http://www.etsi.org/

[TS131.102] “Characteristics of the USIM application”, (ETSI TS 131.102), URL:http://www.etsi.org/

[TS151.011] “Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface”, (ETSI TS 151 011), URL:http://www.etsi.org/

[WAP-219-TLS]	OMA Wireless Public Key Infrastructure V1.0, Open Mobile Alliance™, WAP-219_100-TLS URL:http://www.openmobilealliance.org/
[WBXML1.1]	"WAP Binary XML Content Format Specification", WAP Forum™. SPEC-WBXML-19990616.pdf. URL:http://www.openmobilealliance.org
[WBXML1.2]	"WAP Binary XML Content Format Specification", WAP Forum™. WAP-154-WBXML. URL:http://www.openmobilealliance.org
[WBXML1.3]	"WAP Binary XML Content Format Specification", WAP Forum™. WAP-192-WBXML. URL:http://www.openmobilealliance.org
[WTLS]	"Wireless Transport Layer Security", Open Mobile Alliance™, WAP-261-WTLS, URL:http://www.openmobilealliance.org

2.2 Informative References

[XMLENC]	"XML Encryption Syntax and Processing". W3C. URL:http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[XMLSIGN]	"XML-Signature Syntax and Processing". W3C. URL:http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

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

Authentication Authentication is the process of ascertaining the validity of either the Device or the Device Management Server's identity.

Confidentiality Confidentiality is the ability to keep contents secret from all but the two entities exchanging a message. It does not limit the visibility of the message (being able to eavesdrop), but it does prevent the interpretation of the data being transmitted. Effectively this prevents the contents of a message being understood by anybody but the intended sender and intended recipient.

Content Content means data delivered inside of OMA DM messages <Data>-elements.

Content Trust Content Trust means ability to identify the source of the content.

Credentials Credentials are elements that are required to prove authenticity. Typically a username and a password.

Device The Device is, or is to become managed by one or more remote entities (Device Management Servers). A device may have many characteristics, and many parameters may be made available for reading, writing, deleting and modifying by a Device Management Server.

Device Management Server The Device Management Server is an entity that is responsible for maintaining one or more Devices, in whole or in part. Its role is to facilitate the easy maintenance of a Device.

Integrity Integrity is the ability for a message to maintain its content or at a minimum, have the ability to detect modification or corruption of its content.

Management Session A continuous connection between the Device and the Device Management Server established for the purpose of carrying out one or more device management operations.

Management Trust Management trust means right to manage Device Management Tree in Device.

3.3 Abbreviations

DM	Device Management
MAC	Message Authentication Code
OMA	Open Mobile Alliance
WAP	Wireless Application Protocol

4. Introduction

OMA DM is a protocol based upon SyncML. Its purpose is to allow remote management of any device supporting the OMA DM protocol. Due to the vast range of data needing to be managed on current and future devices, it is necessary to take account of the value of such data. In many situations, the data being manipulated within a device (or being transferred to/from the device) is of high value. In some cases this is confidential data and some degree of protection regarding the confidentiality of that data should be offered. In another case, the integrity of the data being transferred must be maintained, since deliberate or accidental corruption of this data can result in lost revenue or subsequent exploits being facilitated. Finally it's important that both entities (the Device and the Device Management Server) have confidence in the authenticity of the other entity.

5. OMA Device Management Security

5.1 Credentials

Four examples of suitable credentials exchanged between Devices and Device Management servers are shown in the following list.

1. A username (AAUTHNAME in [DMSTDOBJ]) that uniquely identifies the Device Management Server [DMTND] to a Device), a password (AAUTHSECRET in [DMSTDOBJ]) – to be coupled with the username, and a nonce (AAUTHDATA in [DMSTDOBJ]) – to allow for prevention of replay attacks where hashing algorithms are used with static data.

2. A username (AAUTHNAME in [DMSTDOBJ]) that identifies the Device to the Device Management Server, a password (AAUTHSECRET in [DMSTDOBJ]) – to be coupled with username, and a nonce (AAUTHDATA in [DMSTDOBJ]) – to allow for prevention of replay attacks where hashing algorithms are used with static data.

3. A certificate, as specified in [WAP-219-TLS]

4. A network, transport or server specific mechanism, for example, WAP.

For the purpose of Server to Device authentication, if a username, password and nonce are used, the Server MUST use a different password for each client it serves, in order that a client (which possesses a shared secret based on this password) cannot pose effectively as this Server in a interaction with another client.

5.2 Initial Provisioning of Credentials

The initial provisioning of the credentials for a server, so that the Device may be capable of authenticating a specific Device Management Server, is documented in [DMBOOT]. However, other techniques outside of these specifications are not excluded.

Essentially, any suitable technique will deliver at least the bare minimum of information required to establish the DM session. This, of course, includes the Server credential and the Device credential.

5.3 Authentication

Both OMA DM Protocol [DMPRO] client and server MUST be authenticated to each other. Authentication can be performed at different layers. OMA DM servers MUST support both client and server authentication at the transport layer. OMA DM servers MUST request client authentication at the transport layer when transport layer security is requested by the OMA DM client during session establishment. Some clients may not support transport-layer client authentication. Servers MUST authenticate such clients at the application layer. If the transport layer does not have a sufficiently strong authentication feature, OMA DM Protocol layer authentication MUST be used.

Either the client or the server MAY send credentials to each other or challenge the other to send them.

OMA DM clients that do not support client authentication at the transport layer MUST support OMA DM syncml:auth-md5 type authentication. OMA DM clients that support mutual authentication at the transport layer MAY support OMA DM authentication mechanisms such as the syncml:auth-md5 type.The DM server MAY still issue a MD5 challenge when transport layer mutual authentication has already been completed but the session MUST be terminated if the client does not respond with the requested authentication type. The provisioning of credentials/certificates for transport layer authentication is beyond the scope of OMA DM Security.

It is assumed that OMA DM Protocol will often be used on top of a transport protocol that offers session layer authentication so that authentication credentials are exchanged only at the beginning of the session (like in TLS or WTLS). If the transport layer is not able to provide session authentication, however, each request and response MUST be authenticated.

5.3.1 MD-5 authentication in OMA DM

MD-5 authentication [RFC1321] works by supplying primitive `userid:password` in the `Cred` element of the `SyncHdr` as shown below.

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.2</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
      <LocName>Bruce1</LocName>    <!-– userid -->
    </Source>
    <Cred>
      <Meta>
        <Type xmlns="syncml:metinf">syncml:auth-md5</Type>
        <Format xmlns=”syncml:metinf”>b64</Format>
      </Meta>
      <Data>18EA3F……</Data>
          <!-- base64 formatting of MD-5 Digest -->
    </Cred>
    <Meta>
      <MaxMsgSize xmlns="syncml:metinf">5000</MaxMsgSize>
    </Meta>
  </SyncHdr>
         <!–- regular body information here -->
  <SyncBody>
  </SyncBody>
</SyncML>
```

5.3.2 Computation of the MD-5 Digest

The digest supplied in the `Cred` element is computed as follows:

Let H = the MD5 Hashing function.
Let Digest = the output of the MD5 Hashing function.

Let B64 = the base64 encoding function.

Digest = H(B64(H(username:password)):nonce)

This computation allows the authenticator to authenticate without having knowledge of the password. The password is neither sent as part of the `Cred` element, nor is it required to be known explicitly by the authenticator, since the authenticator need only store a pre-computed hash of the username:password string.

5.3.3 Password and nonce usage

Both password and nonce are recommended to be at least 128 bits (16 random octets) in length.

The nonce value MUST be issued in a challenge from either the Device or the Device Management Server. In the case of the credentials being sent prior to a challenge being issued, then the last nonce used shall be reused. The authenticator must be aware that the issuer of the credentials may be using a stale nonce (that is to say, a nonce that is invalid due to some previous communications failure or a loss of data). Because of this, if authentication fails, one more challenge, along with the supply of a new nonce, MUST be made.

A new nonce SHOULD be used for each new session. The sequence of nonce values (as seen across sessions) SHOULD be difficult to predict.

5.3.4 Challenges from non-authenticated agents

In some scenarios, it might be necessary for client and server to accept challenges from agents that have not yet been successfully authenticated. For example, consider the case in which both client and server have outdated nonces, and MD5 or HMAC authentication is used. If they both discard the `Chal` element, they will not have a chance to update their nonce and they will never be able to authenticate each other. To avoid this situation it is RECOMMENDED that client and server use the latest received nonce to build the content of the `Cred` element, even when the nonce is received from a non-authenticated agent. It is also RECOMMENDED that client and server should not over-write the stored copy of the next nonce with one received from a non-authenticated agent, as that would allow malicious agents to replace good nonces with bad ones.

5.4 Integrity

Integrity of OMA DM messages is achieved using a HMAC-MD5 [RFC2104].

This is a Hashed Message Authentication Code that MUST be used on every message transferred between the Device and the Device Management Server (if requested to do so by either entity). The use of integrity checking is OPTIONAL.

5.4.1 How integrity checking is requested

Integrity checking is requested in the same way, and at the same time as authentication challenges in [DMPRO]. A challenge issued for `syncml:auth-MAC` will use the same `Meta` data for `Type`, `Format`, and `NextNonce` as `syncml:auth-md5`. A new authentication type, `syncml:auth-MAC`, may be requested by either the client or the management server (or simply supplied prior to a challenge ever being issued). When used, this authentication type MUST be specified in the transport header and MUST NOT be specified using the `Cred` element.

Note that the recipient of a challenge MUST respond with the requested authentication type, else the session MUST be terminated. For example, a challenge requesting the HMAC engenders a reply with valid Basic Authentication credentials, the session will be terminated despite the validity of the authentication credentials that were actually supplied.

5.4.2 How the HMAC is computed

The HMAC is computed as described below, and uses MD-5 as its hashing function. The HMAC relies upon the use of a shared secret (or key), which in this application it itself a hash.

The HMAC value MUST be computed by encoding in base64 the result of the digest algorithm applied as follows:

H(B64(H(username:password)):nonce:B64(H(message body)))

where H(X) is the result of the selected digest algorithm (MD-5) applied to octet stream X, and B64(Y) is the base64 encoding of the octet stream Y.

5.4.3 How the HMAC is specified in the OMA DM message

The HMAC itself MUST be transported along with the original OMA DM message. This is achieved by inserting the HMAC into a transport header called `x-syncml-hmac`. This technique works identically on HTTP, WAP, and OBEX. The HMAC is calculated initially by the sender using the entire message body, either in binary form (WBXML) or text form (XML). The receiver applies the same technique to the incoming message.

The header `x-syncml-hmac` contains multiple parameters, including the HMAC itself, the user or server identifier, and an optional indication of which HMAC algorithm is in use. (The only one currently defined is MD-5).

The value of the `x-syncml-hmac` header is defined as a comma separated list of attribute-values pairs. The rule "#rule" and the terms "token" and "quoted-string" are used in accordance to their definition in the HTTP 1.1 specifications [RFC2616].

Here is the formal definition:

```
syncml-hmac = #syncml-hmac-param
```

where:

```
syncml-hmac-param = (algorithm | username | mac)
```

The following parameters are defined:

```
algorithm = "algorithm" "=" ("MD5" | token)

username = "username" "=" username-value

mac = "mac" "=" mac-value
```

where:

```
username-value = quoted-string

mac-value = base64-string
```

The parameter `algorithm` can be omitted, in that case MD5 is assumed. The parameter `username` MUST be specified. The parameter `mac` MUST be specified.

Note that a `base64-string` is any concatenation of the characters belonging to the base64 Alphabet, as defined in [RFC1521].

Example:

```
x-syncml-hmac: algorithm=MD5, username="Robert Jordan",
   mac=NTI2OTJhMDAwNjYxODkwYmQ3NWUxN2RhN2ZmYmJlMzk
```

The `username-value` is the identical string from the `LocName` of the `Source` element of the `SyncHdr`, and represents the identity of the sender of the message. The presence of the username in the message header allows the calculation and validation of the HMAC to be independent of the parsing of the message itself.[1]

Upon receiving a message, the steps are:

1. Check for the HMAC in the message header; extract it and the username.
2. Using the `username`, look up the secret key from storage. This key is itself a hash, which incorporates the username and password, as described earlier.
3. Either parse the message;
4. Or, validate the digest.

In either sequence of steps, the digest is calculated based on the entire message body, which is either a binary XML document (WBXML) or a text XML document.

[1] The independence established between the validation of the HMAC and the parsing of the message permits these operations to be performed in any order, or even in parallel. And, if in the future SyncML allows a simpler method of constructing a response indicating that authentication failed, it will be possible to issue this response without ever spending the time needed to parse the message itself.

After the HMAC is computed by the receiver (if it was present), the supplied HMAC and the computed HMAC can be compared in order to establish the authenticity of the sender, and also the integrity of the message. If the HMAC was expected (e.g. if a challenge for it had been issued) and either it or the userid are not supplied in the correct transport header, then an authentication failure results (as if they had been supplied, and were incorrect).

Once the HMAC technique is used, it MUST be used for all subsequent messages until the end of the OMA DM session. The Status code sent back for the SyncHdr MUST be 200 to indicate authenticated for this message. In addition, the `NextNonce` element MUST be sent and used for the next HMAC credential check. Failure to meet these requirements MUST result in a termination of the session.

5.4.4 HMAC and nonce value

A new nonce MUST be used for every message. The new nonce will be obtained via the NextNonce value in the previous message. In addition, since HMAC credentials MUST be verified for each message, the SyncHdr status code for an authenticated message MUST be 200.

5.4.5 Use of transport protocols providing authentication and integrity

Note that the static conformance requirements for the HMAC feature is independent of its use. Neither client nor server need supply the HMAC, unless challenged for it. For example, if it is deemed that an already authenticated transport protocol connection has already been established, then the Device or the Device Management Server MAY choose not to authenticate. In this particular situation, neither server nor client is expected to issue a challenge for it. According to the general techniques specified in [DMPRO], a DM client that supports mutual authentication at the transport layer MAY choose not to support OMA DM authentication mechanisms. In this particular case, the server MAY still issue a HMAC challenge, but the session MUST end if the client does not respond with the requested authentication type. The use of transport layer protocols is specified further in Section 5.5.1.1. The provisioning of credentials/certificates for transport layer authentication however, is beyond the scope of OMA DM Security.

5.5 Confidentiality

Confidentiality in OMA DM has two major aspects; the confidentiality of information being transferred over a transport protocol, and the confidentiality of information between Device Management Servers.

5.5.1 Confidentiality of information during transport

Currently there is no inbuilt ability for the OMA DM protocol itself to provide confidentiality of the data being transferred between the Device and the Device Management Server. However, there are a number of techniques that OMA DM is compatible with that do provide this ability:

5.5.1.1 Transport protocols that support encryption

The use of a transport layer protocol that supports encryption is RECOMMENDED for use where the exposure of the data to third party could have significantly negative consequences.. Note that it is possible to use transports, which give confidentiality, without also having authentication. In these cases, confidentiality may be at risk.

When using OMA DM over HTTP:

- The device management server MUST support both TLS 1.0 [TLS] and SSL3.0 [SSL3.0]
- The device management server MUST use TLS 1.0 or SSL3.0
- The device management client MUST use TLS 1.0 or SSL3.0
- The device management client MUST identify that the device management server is using TLS1.0 or SSL3.0
- A device management session SHALL NOT take place over SSL2.0 or less.
- The device management server MUST support all of the following cipher suites, all of which provide authentication, confidentiality and integrity, when using a TLS1.0 session

  - TLS_RSA_WITH_AES_128_CBC_SHA-1
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_RC4_128_SHA
- The device management client MUST support at least one of the following cipher suites, all of which provide authentication, confidentiality and integrity, when using a TLS1.0 session
  - TLS_RSA_WITH_AES_128_CBC_SHA-1
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_RC4_128_SHA
- The device management 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 device management 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 device management server MAY accept the usage of other cipher suites with at least 128 bit symmetric keys when using an SSL3.0 or TLS1.0 session.
- The device management 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 device management 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

5.5.1.2 Management object encryption

OMA DM fully supports the use of encrypted management objects, which may remain encrypted within the Device Management tree, or be decrypted upon receipt by the Device or Device Management Server.

Depending upon implementation, an object may be encrypted prior to transmission over a non encrypted transport layer, and remain encrypted in storage space within either the Device Management Server or the Device, or, it may be decrypted immediately after receipt, and stored internally in unencrypted format.

No restrictions are placed upon the encryption technique used, since this is independent of the OMA DM protocol itself.

5.5.2 Confidentiality of information between Device Management Servers

OMA DM offers the ability for a Device Management Server to make private any data that is stored under Device Management control from another Device Management Server. This is facilitated by the use of an ACL (Access Control List) that allows the protection of any group, or any individual Device Management object.

5.5.2.1 The Access Control List

The Access Control List allows a hierarchical assignment of Access Rights based upon Device Management Server Identifiers's (Unique identifiers for the Device Management Servers [DMTND]). A detailed description of the ACL can be found in [DMTND].

5.6 Notification Initiated Session

OMA DM offers the ability for a Device Management Server to make a request to a Device to establish a Management Session. The security of this message depends upon a digest. The specification of this message can be found in [DMNOTI].

5.7 Security for Bootstrap Operation

Bootstrapping is a sensitive process that may involve communication between two parties without any previous relationship or knowledge about each other. In this context, security is very important. The receiver of a bootstrap message needs to know that the information originates from the correct source and that it has not been tampered with en-route. It is important that DM clients accept bootstrapping commands only from authorized DM or CP servers.

5.7.1 Bootstrap via CP

The CP bootstrap mechanism is defined in [PROVBOOT].

5.7.1.1 Smartcard

The CP Bootstrap mechanism from the smartcard is defined in [PROVSC].

5.7.2 Bootstrap via DM

5.7.2.1 HMAC Computation for Bootstrap

The HMAC is calculated in the following way:

First, the bootstrap document is encoded in the WBXML format [WBXML1.1], [WBXML1.2], [WBXML1.3]. The encoded document and the shared secret are then input as the data and key, respectively, for the HMAC calculation [RFC2104], based on the SHA-1 algorithm [SHA], as defined in the WTLS specification [WTLS]. The output of the HMAC (M = HMAC-SHA(K, A)) calculation is encoded as a string of hexadecimal digits where each pair of consecutive digits represent a byte. The hexadecimal encoded output from the HMAC calculation is then included in the security information.

The security method and HMAC are then passed as parameters to the content type in the format like this:

Content-Type: MIME type; SEC=type; MAC=digest

Where:

MIME type is application/vnd.syncml.dm+wbxml (cannot use XML for bootstrap)

SEC = “NETWORKID”, “USERPIN”, or “USERPIN_NETWORKID”. Other types may also be used.

Digest is the computed HMAC value as stated above.

5.7.2.2 Transports

Since any transport may be used to send the Bootstrap message to the DM client, appropriate security for bootstrapping a device securely MUST be employed. If the transport has this appropriate security, it MUST be employed, otherwise, transport neutral security MUST be employed.

Transport specific security is documented in the transport binding documents.

5.7.2.3 Transport Neutral Security

The following subsections show some methods of transport neutral security. While the server and client MUST support NETWORKID and USERPIN, they are not limited to just those – other methods may be used as long as they employ a level of security appropriate for bootstrap. The combined security of the secret (e.g., randomness, difficulty of obtaining, etc.), the transport and the environment of use should be among the considerations when a bootstrapping service is being implemented.

5.7.2.3.1 NETWORKID

This method relies on some kind of shared secret that the device and the network provider both know before the bootstrap process starts. This could be things like IMSI (for GSM) or ESN (for CDMA). What the shared secret actually is depends on the network provider and the particular device. One advantage with this method is that is can be used without user intervention.

The NETWORKID method requires:

A HMAC value to be calculated using this shared secret and the DM bootstrap message, to be sent along with the message. See section 5.7.2.1.

The protocol used to send the bootstrap message must be capable of transporting both the HMAC value and the OMA DM bootstrap package.

The security type SHALL be specified as ”NETWORKID”.

OMA DM compliant devices and servers MUST support the NETWORKID method.

5.7.2.3.2 USERPIN

This method relies on a PIN that must be communicated to the user out-of-band, or agreed to before the bootstrap process starts.

The USERPIN method requires:

A HMAC value to be calculated using this shared secret and the DM bootstrap message, to be sent along with the message. See section 5.7.2.1.

The protocol used to send the bootstrap message must be capable of transporting both the HMAC value and the OMA DM bootstrap package.

The security type SHALL be specified as ”USERPIN”.

OMA DM compliant devices and servers MUST support the USERPIN method.

5.7.2.3.3 USERPIN_NETWORKID

This is a combination of the NETWORKID and USERPIN methods. It requires the use of a secret shared between the network provider and the device and a user PIN.

The USERPIN_NETWORKID method requires:

A HMAC value to be calculated using this PIN combined with the secret shared between the network provider and the device (with the PIN and secret combined as “PIN:secret”) and the bootstrap message, to be sent with the message. See section 5.7.2.1.

The protocol used to send the bootstrap message must be capable of transporting both the HMAC value and the OMA DM bootstrap package.

The security type SHALL be specified as ”USERPIN_NETWORKID”.

OMA DM compliant devices and servers MAY support the USERPIN_NETWORKID method.

5.7.2.4 Smartcards

While not a transport, per se, smartcards allow for a very secure delivery of bootstrap information.

Smartcard is a generic name for a set of specific specifications: [GSM11.11], [TS151.011], [TS102.221], [TS131.102], [C.S0023-B_v1.0].

Bootstrap data MAY be stored on the smartcard. The behaviour of a DM Client regarding bootstrap data is specified in [DMBOOT].

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-DM_Security-V1_2	09 Feb 2007	Status changed to Approved by TP TP Doc ref# OMA-TP-2007-0075R03-INP_ERP_DM_V1.2_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for DM Client

Item	Function	Reference	Status	Requirement
DM-SEC-C-001	Client must authenticate itself to a server	Section 5.3	M	DM-SEC-C-003 OR DM-SEC-C-007
DM-SEC-C-002	Client must authenticate a server	Section 5.3	M	DM-SEC-C-003 OR DM-SEC-C-007
DM-SEC-C-003	Support for transport layer authentication	Section 5.3	O	
DM-SEC-C-004	Support for HTTP transport	Section 5.5.1.1	O	DM-SEC-C-014 OR DM-SEC-C-15
DM-SEC-C-005	Send credentials to server	Section 5.3	O	
DM-SEC-C-006	Challenge Server	Section 5.3	O	
DM-SEC-C-007	Support for application layer authentication	Section 5.3	O	DM-SEC-C-008 AND DM-SEC-C-010
DM-SEC-C-008	Support for OMA DM syncml:auth-md5 type authentication	Section 5.3	O	
DM-SEC-C-009	Accept challenges from server that has not yet been successfully authenticated	Section 5.3.4	O	
DM-SEC-C-010	Integrity checking using HMAC-MD5	Section 5.4	O	DM-SEC-C-011 AND DM-SEC-C-012
DM-SEC-C-011	Inserting HMAC in transport	Section 5.4.3	O	
DM-SEC-C-012	Using HMAC for all subsequent messages	Section 5.4.3	O	
DM-SEC-C-013	Identifying that the server is using TLS1.0 or SSL3.0	Section 5.5.1.1	O	
DM-SEC-C-014	Support for TLS	Section 5.5.1.1	O	DM-SEC-C-016
DM-SEC-C-015	Support for SSL 3.0	Section 5.5.1.1	O	DM-SEC-C-017
DM-SEC-C-016	Supporting at least one of the cipher suites TLS_RSA_WITH_AES_128_CBC_SHA-1, TLS_RSA_WITH_3DES_EDE_CBC_SHA and TLS_RSA_WITH_RC4_128_SHA	Section 5.5.1.1	O	
DM-SEC-C-017	Support for at least one of SSL_RSA_WITH_RC4_128_SHA and SSL_RSA_WITH_3DES_EDE_CBC_SHA	Section 5.5.1.1	O	

Item	Function	Reference	Status	Requirement
DM-SEC-C-018	Bootstrap Security for Bootstrap via DM Profile	Section 5.7.2.2	O	DM-SEC-C-019 OR DM-SEC-C-020
DM-SEC-C-019	Transport neutral security for Bootstrap via DM Profile	Section 5.7.2.2.	O	DM-SEC-C-021 OR DM-SEC-C-022 OR DM-SEC-C-023
DM-SEC-C-020	Transport layer security for Bootstrap via DM Profile	Section 5.7.2.2.	O	
DM-SEC-C-021	Use of NETWORKID and USERPIN when Bootstrapping via DM Profile	Section 5.7.2.3	O	
DM-SEC-C-022	Support of NETWORKID in Bootstrap via DM Profile	Section 5.7.2.3	O	
DM-SEC-C-023	Support of USERPIN in Bootstrap via DM Profile	Section 5.7.2.3	O	

B.2 SCR for DM Server

Item	Function	Reference	Status	Requirement
DM-SEC-S-001	Different password for each client	Section 5.1	M	
DM-SEC-S-002	Support for client authentication at the transport layer	Section 5.3	M	
DM-SEC-S-003	Send credentials to client	Section 5.3	M	
DM-SEC-S-004	Challenge Client	Section 5.3	O	
DM-SEC-S-005	Support for clients authentication at the application layer	Section 5.3	M	DM-SEC-S-006 AND DM-SEC-S-009 AND DM-SEC-S-010
DM-SEC-S-006	MD5 challenge to client	Section 5.3	O	
DM-SEC-S-007	MD5 challenge to client in conjunction with transport layer security	Section 5.3	O	
DM-SEC-S-008	Supply of a new nonce with one more challenge if authentication fails	Section 5.3.3	M	
DM-SEC-S-009	Using new nonce for each new session	Section 5.3.3	O	
DM-SEC-S-010	Accept challenges from clients that have not yet been successfully authenticated	Section 5.3.4	O	
DM-SEC-S-011	Integrity checking using HMAC-MD5	Section 5.4	O	DM-SEC-S-012 AND DM-SEC-S-013

Item	Function	Reference	Status	Requirement
DM-SEC-S-012	Inserting HMAC in transport	Section 5.4.3	O	
DM-SEC-S-013	Using HMAC for all subsequent messages	Section 5.4.3	O	
DM-SEC-S-014	Support for HTTP transport	Section 5.5.1.1	O	DM-SEC-S-015 AND DM-SEC-S-016
DM-SEC-S-015	Support for TLS 1.0 [TLS]	Section 5.5.1.1	O	
DM-SEC-S-016	Support for SSL3.0 [SSL3.0]	Section 5.5.1.1	O	
DM-SEC-S-017	Using OMA DM over HTTP	Section 5.5.1.1	O	DM-SEC-S-018 OR DM-SEC-S-019
DM-SEC-S-018	Using TLS	Section 5.5.1.1	O	DM-SEC-S-020
DM-SEC-S-019	Using SSL3.0	Section 5.5.1.1	O	DM-SEC-S-021
DM-SEC-S-020	Supporting all three cipher suites TLS_RSA_WITH_AES_128_CBC_SHA-1, TLS_RSA_WITH_3DES_EDE_CBC_SHA and TLS_RSA_WITH_RC4_128_SHA	Section 5.5.1.1	O	
DM-SEC-S-021	Support for both of SSL_RSA_WITH_RC4_128_SHA and SSL_RSA_WITH_3DES_EDE_CBC_SHA	Section 5.5.1.1	O	
DM-SEC-S-022	Bootstrap Security for Bootstrap via DM Profile	Section 5.7.2.2	O	DM-SEC-S-023 OR DM-SEC-S-024
DM-SEC-S-023	Transport neutral security for Bootstrap via DM Profile	Section 5.7.2.2.	O	DM-SEC-S-025 or DM-SEC-S-026 or DM-SEC-S-027
DM-SEC-S-024	Transport layer security for Bootstrap via DM Profile	Section 5.7.2.2.	O	
DM-SEC-S-025	Use of NETWORKID and USERPIN when Bootstrapping via DM Profile	Section 5.7.2.3	O	
DM-SEC-S-026	Support of NETWORKID in Bootstrap via DM Profile	Section 5.7.2.3	O	
DM-SEC-S-027	Support of USERPIN in Bootstrap via DM Profile	Section 5.7.2.3	O	

Appendix C. Security support when Management Trust and Content Trust are separated (Informative)

C.1 Content signing and encryption introduction

There are situations when Management trust and Content Trust need to be separated. The idea of the content signing and encryption is that the Device could identify the source of the Content even when the delivery is done by using some other Device Management Server. Also Content may be encrypted in a way that only the receiver is able to decrypt the Content. By Content, in this context, we mean data delivered inside of OMA DM messages <Data>-elements. In this context the other Device Management Server means the Device Management server, which is not controlled by the content creator or content source. Source of the signed or encrypted Content could be almost anything and most probably it does not know anything about the OMA DM protocol. This is why it must be possible to keep the coupling between the OMA DM-protocol and the Source of the Content as loose as possible.

C.2 Content Signature enabling Authenticity and Integrity

XML-signature [XMLSIGN] offers the signature mechanism to achieve Authenticity and Integrity. Because the messaging between the Source of the Content and Terminal is not possible in most of the cases, we need to agree the mandatory algorithms beforehand. The algorithms that must be supported for Authenticity and Integrity are RSA and SHA-1 as specified in [XMLSIGN]. XML Signature has three ways of representing signature in a document viz: enveloping, enveloped and detached. Enveloped or enveloping signatures are over data within the same XML document as the signature; detached signatures are over data external to the signature element. The use of the "detached" signature is recommended. The format value used for XML-signature data is xml.

XML Signatures are applied to arbitrary digital content (data objects) via an indirection. Data objects are digested, the resulting value is placed in an element (with other information) and that element is then digested and cryptographically signed. XML digital signatures are represented by the Signature element which has the following structure (where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; and "*" denotes zero or more occurrences):

```
<Signature ID?>
          <SignedInfo>
        <CanonicalizationMethod/>
        <SignatureMethod/>
        (<Reference URI? >
            (<Transforms/>)?
            <DigestMethod/>
            <DigestValue/>
        </Reference>)+
    </SignedInfo>
    <SignatureValue/>
    (<KeyInfo/>)?
    (<Object ID?>)*
</Signature>
```

Each resource to be signed has its own <Reference> element identified by the URI attribute.

Rules for XML-signature elements used for enveloping XML-signature [XMLSIGN] in OMA DM Content signature context:

- Content (data), which is to be signed, should be placed after the signature element, if detached signature is being used. This is the recommended way to place the content. In this case the <Reference> element may not contain any URI attribute. In this case The Device must implicitly know the location of the Content
- Content (data), which is to be signed, may be placed inside of <Object> element when enveloping signature is being used.
- <Object> element must not contain any other elements than Content signed and <Object> element must not exist when detached signature is used.
- <Reference> element may not contain any attributes.
- <Reference> element must have child elements <Transforms>, <DigestMethod> and <DigestValue> elements.
- <DigestValue> element contents must be encoded using base64.
- <SignatureValue> element contents must be encoded using base64.
- <Transforms> element must not have <Xpath> child element
- <Signature> element must be a child of <Data> element.
- <KeyInfo> may be included in <Signature> for receiver to verify signature.
- The digest value (in <DigestValue>) is encrypted with sender's private key to produce <SignatureValue>. The receiver then decrypts the signature with the sender's public key (in KeyInfo/KeyValue) to produce digest value (which sender computed), This hash value is compared to the digest value computed by the receiver.

Example of OMA DM message with signed content (recommended, detached signature method):

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    …
  </SyncHdr>
  <SyncBody>
    …
    <Replace>
      <CmdID>4</CmdID>
      <Meta>
        <Format xmlns="syncml:metinf">xml</Format>
        <Type xmlns="syncml:metinf">application/xml</Type>
      </Meta>
      <Item>
        <Target>
          <LocURI>./my_mgmt_obj/file</LocURI>
        </Target>
        <Data>
          <![CDATA[
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
              <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              <SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
            sha1"/>
              <Reference>
                <Transforms>
                  <Transform
```

```
                    Algorithm="http://www.w3.org/2001/10/xml-exc-
            c14n#"/>
                </Transforms>
                <DigestMethod

            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue> LyLsF094hPi4wPU... </DigestValue>
              </Reference>
            </SignedInfo>
            <SignatureValue>
              Hp1ZkmFZ/2kQLXDJbchm5gK...
            </SignatureValue>
            <KeyInfo>
              <KeyValue xmlns="http://www.w3.org/2000/09/xmldsig#">
                . . .
              </KeyValue>
            </KeyInfo>
          </Signature>
          ]]>
          MY_SIGNED_BINARY_OR_XML_CONTENT...
        </Data>
      </Item>
    </Replace>
  </SyncBody>
</SyncML>
```

Example of OMA DM message with signed content (enveloping signature method):

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    …
  </SyncHdr>
  <SyncBody>
    …
    <Replace>
      <CmdID>4</CmdID>
      <Meta>
        <Format xmlns="syncml:metinf">xml</Format>
        <Type xmlns="syncml:metinf">application/xml</Type>
      </Meta>
      <Item>
        <Target>
          <LocURI>./my_mgmt_obj/file</LocURI>
        </Target>
        <Data>
          <![CDATA[
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
              <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              <SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
              <Reference>
                <Transforms>
                  <Transform
```

```
                    Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                </Transforms>
                <DigestMethod

Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue> LyLsF094hPi4wPU... </DigestValue>
              </Reference>
            </SignedInfo>
            <SignatureValue>
              Hp1ZkmFZ/2kQLXDJbchm5gK...
            </SignatureValue>
            <KeyInfo>
              <KeyValue xmlns="http://www.w3.org/2000/09/xmldsig#">
                . . .
              </KeyValue>
            </KeyInfo>
            <Object>
              ASDFASDFASDFASDG...
            </Object>
          </Signature>
          ]]>
        </Data>
      </Item>
    </Replace>
  </SyncBody>
</SyncML>
```

Example of OMA DM message with signed content (enveloped signature method):

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    …
  </SyncHdr>
  <SyncBody>
    …
    <Replace>
      <CmdID>4</CmdID>
      <Meta>
        <Format xmlns="syncml:metinf">xml</Format>
        <Type xmlns="syncml:metinf">application/xml</Type>
      </Meta>
      <Item>
        <Target>
          <LocURI>./my_mgmt_obj/file</LocURI>
        </Target>
        <Data>
          <![CDATA[
          <MyObject ID=MY_ID>
            <MY_XML_CONTENT_HEADER />
            <MY_XML_CONTENT_DATA />
          </MyObject>
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
              <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              <SignatureMethod
```

```
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
              <Reference>
                <Transforms>
                  <Transform
                    Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                </Transforms>
                <DigestMethod

Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue> LyLsF094hPi4wPU... </DigestValue>
              </Reference URI="#MY_ID">
            </SignedInfo>
            <SignatureValue>
              Hp1ZkmFZ/2kQLXDJbchm5gK...
            </SignatureValue>
            <KeyInfo>
              <KeyValue xmlns="http://www.w3.org/2000/09/xmldsig#">
                . . .
              </KeyValue>
            </KeyInfo>
          </Signature>
          ]]>
        </Data>
      </Item>
    </Replace>
  </SyncBody>
</SyncML>
```

C.3 Content Confidentiality

XML-encryption [XMLENC] offers the encryption mechanism to achieve Content Confidentiality. Because the messaging between the Source of the Content and Terminal is not possible in most of the cases, we must agree the mandatory algorithms beforehand. The algorithms that must be supported for Confidentiality are RSA and AES128 as specified in [XMLENC]. MIME type for XML-encryption data is application/xenc+xml.

If content is signed and encrypted the signature must be done first and the encryption must be placed over the entire signed content.

Rules for XML-encryption elements used for XML-encryption [XMLENC] in OMA DM Content Encryption context:

- XML-Encryption tree must be placed as a child of a <Data> element (whose content we want to encrypt) in <SyncBody>.
- OMA DM Content must be encrypted using a symmetric key AES128, i.e. outer <EncryptionMethod> element must have algorithm attribute set to a symmetric keying method.
- Symmetric key must be encrypted by an asymmetric key RSA-1_5, i.e. inner <EncryptionMethod> element must have algorithm attribute set to an asymmetric keying method (i.e. receiver's public key).
- <KeyInfo> must be included in <EncryptedData> and in <EncryptedKey> for receiver to inform encryption keys.

Example of OMA DM message with encrypted content:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    …
  </SyncHdr>
  <SyncBody>
    …
    <Replace>
      <CmdID>3</CmdID>
      <Meta>
        <Format xmlns="syncml:metinf">xml</Format>
        <Type xmlns="syncml:metinf">application/xenc+xml</Type>
      </Meta>
      <Item>
        <Target>
          <LocURI>./my_mgmt_obj/file</LocURI>
        </Target>
        <Data>
          <![CDATA[
            <xenc:EncryptedData
              Type=http://www.w3.org/2001/04/xmlenc#Element>
              <EncryptionMethod
                Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc'/>
              <KeyInfo>
              <EncryptedKey>
              <EncryptionMethod
                    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
              <KeyInfo>
                <KeyName>rsaKey</KeyName>
              </KeyInfo>
              <CipherData>
                <CipherValue>
                  xyzabc
                </CipherValue>
                </CipherData>
              </EncryptedKey>
              </KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>...</xenc:CipherValue>
              </xenc:CipherData>
          </xenc:EncryptedData>
          ]]>
        </Data>
      </Item>
    </Replace>
  </SyncBody>
</SyncML>
```