SyncML Device Management Security

Abstract

This document describes the security strategies for SyncML Device Management and details the specific techniques to be used as mandated or recommended by SyncML Device Management.

SyncML Initiative

The following companies are Sponsors of the SyncML Initiative:

Ericsson
IBM
Lotus
Matsushita Communications Industrial Co., Ltd.
Motorola
Nokia
Openwave
Starfish Software
Symbian

Revision History

Revision	**Date**	**Comments**
V 1.1	2002-02-15	First release

Copyright Notice

Table of Contents

1 Introduction

SyncML DM is a protocol based upon SyncML. Its purpose is to allow remote management of any device supporting the SyncML 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.

This document describes the measures and techniques incorporated into SyncML DM to ensure these requirements, and where not facilitated by SyncML DM itself, the recommended protocols that will satisfy the requirement.

2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

Any reference to components of the Device Information DTD or XML snippets are specified in this **`typeface`**.

3 Terminology

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.

Authentication

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

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.

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.

Credentials

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

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.

4 Credentials

The credentials exchanged between Device and Device Management server are basically taken from the following list of items.

1) Server ID (this is a unique ID that identifies the Device Management Server)

2) A username that identifies the Device to the Device Management Server

3) A password – to be coupled with username, or Server ID

4) A nonce – to allow for prevention of replay attacks where hashing algorithms are used with static data (overcoming replay attacks).

For the purpose of Device to Server authentication, a username and password and nonce are required (even if never sent, they are hashed)

For the purpose of Server to Device authentication, a Server ID, password and nonce are required. 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.

All security with SyncML DM is based upon these four credentials.

5 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 [3].
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 is of course includes the Server ID, password and nonce, in addition to the server address.

6 Authentication

Authentication within SyncML DM uses the same general technique as per SyncML[7]. Namely, either the client or the server MAY send credentials to each other or challenge the other to send them.

For SyncML DM, the `syncml:auth-MD5` type MUST be supported.

6.1 MD-5 authentication in SyncML DM

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

```
<SyncML>
  <SyncHdr>
    <VerDTD>1.1</VerDTD>
    <VerProto>DM/1.1</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
    <Cred>
      <Meta>
        <Type xmlns="syncml:metinf">syncml:auth-MD5</Type>
      </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>
```

6.2 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.

6.3 Password and nonce usage

Both password and nonce are recommended to be at least 128 bits (6 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.

7 Integrity

Integrity of SyncML DM messages is achieved using a HMAC-MD5 (see 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).

Both the Device Management Server and the Device MUST support the use of integrity checking as detailed below.

The use of integrity checking is OPTIONAL.

7.1 How Integrity checking is requested

Integrity checking is requested in the same way, and at the same time as authentication challenges in [7].

A new authentication type, `syncml:auth-MAC,` may be requested by either the Device or the Device Management Sever (or simply supplied prior to a challenge ever being issued).

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.

7.2 How the HMAC is computed

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

The HMAC value should 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.

7.3 How the HMAC is specified in the SyncML DM message

The HMAC itself must also be transported along with the original SyncML 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 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 [10].

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 [8].

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.

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).

[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.

Once the HMAC technique is used, it MUST be used for all subsequent messages until the end of the SyncML DM session. Failure to do so MUST result in a termination of the session.

7.4 HMAC and nonce value

The effect of the nonce relies on the `MsgID` (message identifier), which acts as a counter with a value that is incremented with each new message. Because the HMAC algorithm digests the nonce and the `MsgID` (along with the message), the effect of a changing nonce with each new message is achieved.

7.5 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. The provisioning of credentials / certificates for transport layer authentication is beyond the scope of SyncML DM Security.

8 Confidentiality

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

8.1 Confidentiality of information during transport.

Currently there is no inbuilt ability for the SyncML 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 SyncML DM is compatible with that do provide this ability:

8.1.1 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. Transport protocols such as TLS or HTTPS are to be encouraged. Note that it is possible to use these transports, which give confidentiality, without also having authentication. In these cases, confidentiality may be at risk.

8.1.2 Management object encryption

SyncML 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 SyncML DM protocol itself.

8.2 Confidentiality of information between Device Management Servers

SyncML 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.

8.2.1 The Access Control List

The Access Control List allows a hierarchical assignment of Access Rights based upon Device Management Server ID's (Unique identifiers for the Device Management Servers).

A detailed description of the ACL can be found in [2].

9 Notification Initiated Session

SyncML 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 [5].

10 Bootstrap

The specification of this message can be found in [3]

11 References

[1] [RFC2104] HMAC: Keyed-Hashing for Message Authentication, IETF.

[2] [SyncML DM tnd] SyncML Device Management Tree and Description, SyncML.

[3] [SyncML DM boot] SyncML Device Management Bootstrap, SyncML.

[4] [RFC1321] The MD5 Message-Digest Algorithm IETF.

[5] [SyncML DM notification] SyncML Notification Initiated Session, SyncML.

[6] [RFC2119] Key words for use in RFCs to Indicate Requirement Levels, IETF.

[7] [SyncML proto] SyncML Synchronization Protocol Specification, SyncML.

[8] [RFC1521] MIME (Multipurpose Internet Mail Extensions) Part One IETF.

[9] [RFC2617] HTTP Authentication: Basic and Digest Access Authentication IETF.

[10] [RFC2616] Hypertext Transfer Protocol -- HTTP/1.1 IETF.