SyncML Representation Protocol

Approved Version 1.1.2 – 11 Jul 2004

Open Mobile Alliance

OMA-SyncML-RepPro-V1_1_2-20040711-A

Continues the Technical Activities
Originated in the SyncML Initiative

Contents

1. Scope

The SyncML Initiative, Ltd. was a not-for-profit corporation formed by a group of companies who co-operated to produce an open specification for data synchronization and device management. Prior to SyncML, data synchronization and device management had been based on a set of different, proprietary protocols, each functioning only with a very limited number of devices, systems and data types. These non-interoperable technologies have complicated the tasks of users, manufacturers, service providers, and developers. Further, a proliferation of different, proprietary data synchronization and device management protocols has placed barriers to the extended use of mobile devices, has restricted data access and delivery and limited the mobility of the users.

SyncML Components:

- SyncML is a specification that contains the following main components:
- An XML-based representation protocol
- A synchronization protocol and a device management protocol
- Transport bindings for the protocol

The data representation specifies an XML DTD that allows the representation of all the information required to perform synchronization or device management, including data, metadata and commands. The synchronization and device management protocols specify how SyncML messages conforming to the DTD are exchanged in order to allow a SyncML client and server to exchange additions, deletes, updates and other status information.

There are also DTDs which define the representation of information about the device such as memory capacity, and the representation of various types of meta information such as security credentials.

Although the SyncML specification defines transport bindings that specify how to use a particular transport to exchange messages and responses, the SyncML representation, synchronization and device management protocols are transport - independent. Each SyncML package is completely self-contained, and could in principle be carried by any transport. The initial bindings specified are HTTP, WSP and OBEX, but there is no reason why SyncML could not be implemented using email or message queues, to list only two alternatives. Because SyncML messages are self-contained, multiple transports may be used without either the server or client devices having to be aware of the network topology. Thus, a short-range OBEX connection could be used for local connectivity, with the messages being passed on via HTTP to an Internet-hosted synchronization server.

To reduce the data size, a binary coding of SyncML based on the WAP Forum's WBXML is defined. Messages may also be passed in clear text if required. In this and other ways SyncML addresses the bandwidth and resource limitations imposed by mobile devices.

SyncML is both data type and data store independent. SyncML can carry any data type which can be represented as a MIME object. To promote interoperability between different implementations of SyncML, the specification includes the representation formats used for common PIM data.

This document specifies the common XML syntax and semantics used by all SyncML protocols. The SyncML representation protocol is defined by a set of messages that are conveyed between entities participating in a SyncML operation. The SyncML representation protocol embodies the concept of a SyncML Package. The SyncML Package performs some set of operations. This conceptual "package" permits either a "batch" of multiple operations put together in a single SyncML Message or conveyed as separate SyncML Messages, each containing a single operation.

2. References

2.1 Normative References

[DMCONF] "SyncML Device Management Conformance Requirements", Open Mobile Alliance™, OMA-SyncML-DMConf-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[DMPRO] "SyncML Device Management Protocol", Open Mobile Alliance™, OMA-SyncML-DMProtocol-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[DMREPU] "SyncML Representation Protocol, Device Management Usage", Open Mobile Alliance™, OMA-SyncML-DMRep-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[DMSEC] "SyncML Device Management Security", Open Mobile Alliance™, OMA-SyncML-DMSec-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[DSPRO] "SyncML Synchronization Protocol", Open Mobile Alliance™, OMA-SyncML-DataSyncProtocol-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[DSREPU] "SyncML Representation Protocol, Data Synchronization Usage", Open Mobile Alliance™, OMA-SyncML-DataSyncRep-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[IMEI] "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Numbering, addressing and identification" (3G TS 23.003 Version 3.4.0 Release 1999), http://webapp.etsi.org/action/PU/20000523/ts_123003v030400p.pdf

[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

[RFC1321] " The MD5 Message-Digest Algorithm ", R. Rivest, et al., April 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed & N. Borenstein, November 1996, http://www.ietf.org/rfc/rfc2045.txt

[WBXML] "WAP Binary XML Content Format Specification", WAP Forum™, WAP-240-WBXML, URL:http://www.openmobilealliance.org/tech/docs

[XML] "Extensible Markup Language (XML) 1.0", World Wide Web Consortium Recommendation, http://www.w3.org/TR/REC-xml

2.2 Informative References

None.

3. Terminology and Conventions

3.1 Conventions

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

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

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

3.2 Definitions

Application - A SyncML application that supports the SyncML protocol. The application can either be the originator or recipient of the SyncML protocol commands. The application can act as a SyncML client or a SyncML server.

Capabilities exchange - The SyncML capability that allows a client and server to exchange what device, user and application features they each support.

Client - A SyncML Client refers to the protocol role when the application issues SyncML "request" messages. For example in data synchronization, the `Sync` SyncML Command in a SyncML Message.

Command - A SyncML Command is a protocol primitive. Each SyncML Command specifies to a recipient an individual operation that is to be performed. For example, the SyncML Commands supported by this specification include `Add`, `Alert`, `Atomic`, `Copy`, `Delete`, `Exec`, `Get`, `Map`, `Replace`, `Search`, `Sequence` and `Sync`.

Data - A unit of information exchange, encoded for transmission over a network.

Data collection - A data element which acts as a container of other data elements, (e.g., {c {{i_1, $data_1$}, ... {i_n, $data_n$}}}). In SyncML, data collections are synchronized with each other. See data element.

data element -A piece of data and an associated identifier for the data, (e.g., {i, data}).

Data element equivalence -When two data elements are synchronized. The exact semantics is defined by a given data synchronization model.

Data exchange - The act of sending, requesting or receiving a set of data elements.

Data format - The encoding used to format a data type. For example, characters or integers or character encoded binary data.

Data type - The schema used to represent a data object (e.g., text/calendar MIME content type for an iCalendar representation of calendar information or text/directory MIME content type for a vCard representation of contact information).

Data synchronization - The act of establishing an equivalence between two data collections, where each data element in one item maps to a data item in the other, and their data is equivalent.

Data synchronization protocol - The well-defined specification of the "handshaking" or workflow required to accomplish synchronization of data elements on an originator and recipient data collection. The SyncML specification forms the basis for specifying an open data synchronization protocol.

Message - A SyncML Message is the primary contents of a SyncML Package. It contains the SyncML Commands, as well as the related data and meta-information. The SyncML Message is an XML document.

Operation - A SyncML Operation refers to the conceptual transaction achieved by the SyncML Commands specified by a SyncML Package. For example in the case of data synchronization, "synchronize my personal address book with a public address book".

Originator - The network device that creates a SyncML request.

Package - A SyncML Package is the complete set of commands and related data elements that are transferred between an originator and a recipient. The SyncML package can consist of one or more SyncML Messages.

Parser - Refers to an XML parser. An XML parser is not absolutely required to support SyncML. However, a SyncML implementation that integrates an XML parser may be easier to enhance.

This document assumes that the reader has some familiarity with XML syntax and terminology.

Recipient - The network device that receives a SyncML request, processes the request and sends any resultant SyncML response.

Representation protocol - A well-defined format for exchanging a particular form of information. SyncML is a representation protocol for conveying data synchronization and device management operations.

SyncML request message - An initial SyncML Message that is sent by an originator to a recipient network device.

SyncML response message - A reply SyncML Message that is sent by a recipient of a SyncML Request back to the originator of the SyncML Request.

Synchronization data - Refers to the data elements within a SyncML Command. In a general reference, can also refer to the sum of the data elements within a SyncML Message or SyncML Package.

Server - A SyncML Server refers to the protocol role when an application issues SyncML "response" messages. For example in the case of data synchronization, a Results Command in a SyncML Message.

3.3 Abbreviations

URI	Uniform Resource Identifier [RFC2396]
URL	Uniform Resource Locator [RFC2396]
WAP	Wireless Application Protocol
XML	Extensible Markup Language

4. Introduction

This document specifies the common XML syntax and semantics used by all SyncML protocols.

The SyncML representation protocol is defined by a set of messages that are conveyed between entities participating in a SyncML operation. The messages are represented as an XML document. XML is the industry standard for text document mark-up, as defined in [XML].

The SyncML representation protocol also can be identified as a MIME content type. MIME is the Internet standard for identifying multipurpose message contents. It provides a useful mechanism for differentiating between different content and document types.

The SyncML representation protocol supports protocol models that are based on a request/response command structure, as well as those that are based on a "blind push" command structure.

The SyncML representation protocol embodies the concept of a SyncML Package. The SyncML Package performs some set of operations. This conceptual "package" permits either a "batch" of multiple operations put together in a single SyncML Message or conveyed as separate SyncML Messages, each containing a single operation. SyncML Messages are the body of the MIME entities.

5. SyncML

5.1 SyncML Package and Messages

In SyncML, the operations are conceptually bound into a ***SyncML Package***. The SyncML Package is just a conceptual frame for one or more ***SyncML Message***s that are required to convey a set of protocol semantics.

A SyncML Message is a well-formed, but not necessarily valid, XML document. The document is identified by the `SyncML` root or document element type. This element type acts as a parent container (i.e., root element type) for the SyncML Message.

The SyncML Message, as specified before, is an individual XML document. The document consists of a header, specified by the `SyncHdr` element type, and a body, specified by the `SyncBody` element type. The SyncML header specifies routing and versioning information about the SyncML Message. The SyncML body is a container for one or more ***SyncML Command***s. The SyncML Commands are specified by individual element types. The SyncML Commands act as containers for other element types that describe the specifics of the SyncML command, including any data or meta-information.

5.2 SyncML Commands

SyncML defines the following "request" commands:

- `Add`. Allows the originator to ask that a data element or data elements supplied by the originator be added to data accessible to the recipient.
- `Alert`. Allows the originator to notify the recipient. The notification can be used as an application-to-application message or a message intended for display through the recipient's user interface.
- `Atomic`. Allows the originator to indicate that a set of commands to be performed with all or nothing semantics.
- `Copy`. Allows the originator to ask that a data element or data elements accessible to the recipient be copied.
- `Delete`. Allows the originator to ask that a data element or data elements accessible to the recipient be deleted. A `Delete` command can include a request for the archiving of the data.
- `Exec`. Allows the originator to ask that a named or supplied executable is invoked by the recipient.
- `Get`. Allows the originator to ask for a data element or data elements from the recipient. A get can include the resetting of any meta-information that the recipient maintains about the data element or collection.
- `Map`. Allows the originator to ask the recipient to update the identifier mapping between two data collections.
- `Put`. Allows the original to put a data element or data elements on to the recipient.
- `Replace`. Allows the originator to ask that a data element or data elements accessible to the recipient be replaced. This command makes a complete replacement of the data element.
- `Search`. Allows the originator to ask that the supplied query be executed against a data element or data elements accessible to the recipient.
- `Sequence`. Allows the originator to indicate that a set of commands is to be performed in the specified sequence.
- `Sync`. Allows the originator to specify that the included commands should be treated as part of the synchronization of two data collections.

SyncML defines the following "response" commands:

- `Status`. Indicates the completion status of an operation or that an error occurred while processing a previous request.
- `Results`. Used to return the data results of either a `Get` or `Search` SyncML Command.

The SyncML Commands themselves do not fully define the semantics of the SyncML Operation. For example, "Adding" a document to an application to a database may have very different semantics from "Adding" a transaction request to a queue. The semantics of a SyncML Operation are determined by the type of data that is being operated upon. This means that it is

possible for an originator to request an operation of a particular recipient that makes no sense to the recipient. In that case, the recipient MUST return an error response status code.

5.3 Security

An objective of SyncML is to provide a framework for secure operation. SyncML itself does not define any new security schemes. Instead, it provides the framework to challenge authentication, authentication, authorization and inclusion of encrypted data in a SyncML Package. In addition, the originator and recipient may use the security mechanisms of the underlying transport to authenticate each other and to provide a secure transport for the exchange of SyncML Packages.

SyncML can be used by an originator to encapsulate authentication information in the `Cred` element type. Implementations conforming to this specification MUST support the "Basic" and "MD5 Digest" schemes.

SyncML can also be used to allow an originator to challenge the authentication of a recipient with the `Chal` element type. Not all authentication schemes provide a challenge mechanism. However, the MD5 Digest scheme does provide such a capability.

The Basic scheme is identified by the URI `syncml:auth-basic`. This authentication scheme is a Base64 character encoding, as defined by Section 6.8, "Base64 Content-Transfer-Encoding" in [RFC2045], of the concatenation of the originator's userid, followed by the COLON (i.e., ":") separator character, followed by the password associated with the specified userid. This authentication scheme is susceptible to the threat of network eavesdrop, but is simple to implement. However, care should be taken when using this scheme. For example, a user is strongly advised to consider using additional security considerations, such as an encrypted transport connection.

The MD5 Digest scheme is identified by the URI `syncml:auth-md5`. Let MD5(data) denote the result of applying the MD5 hash algorithm to “data”, the result is a 128-bit binary quantity. Let A be the concatenation of an authentication identifier as the originator’s userid, followed by the COLON (i.e., “:”) separator character, followed by some secret known by the originator and recipient such as the originator's password for the corresponding userid, for instance:

```
A=”Bruce1:OhBehave”

Let AD be defined as:

AD = MD5(A)
```

Let B64(data) denote the result of the base64 encoding algorithm applied to “data”. This authentication scheme is the MD5 digest form of the concatenation of B64(AD), followed by the COLON (i.e. “:”) separator character, followed by the recipient specified nonce string. The maximum duration that the nonce string can be used by the originator is the current SyncML session. Note that issuing a nonce does not constitute use – a nonce may be issued for use in the next session. More frequent changes to the nonce string can be specified with the `NextNonce` element type within the `Meta` element type of the `Chal` element type. The MD5 digest algorithm and a publicly available source code for generating MD5 digest strings is specified by [RFC1321]. The MD5 credential, a 128-bit binary digest value, MUST be Base64 character encoded when transferred as clear-text XML. For WBXML representation, the additional Base64 character encoding is not necessary.

Other authentication schemes can be specified by prior agreement between the originator and the recipient.

The authentication procedures for the SyncML Data Synchronization protocol are defined in [DSPRO]. The authentication procedures for the SyncML Device Management protocol are defined in [DMSEC].

To specify the userid for the credentials, when the credentials do not include it in the resolvable form, the userid MUST be transferred in the `LocName` element of `Source` in `SyncHdr`. If the userid can be resolved from the credentials, e.g., in the case of the Basic authentication, it can be omitted from the `LocName` element to reduce the number of bytes to be transferred.

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>
```

```
    <VerDTD>1.1</VerDTD>
    <VerProto>SyncML/1.1</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/sync-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
      <LocName>Bruce2</LocName> <!-- userid -->
    </Source>
    <Cred>
      <Meta>
          <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
           <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
        <!— Base64 coded MD5 digest, for user “Bruce2”, password
“OhBehave”, nonce
         “Nonce” " -->
    </Cred>
  </SyncHdr>
  <SyncBody>...</SyncBody>
</SyncML>
```

5.3.1 Optional Authentication Types

SyncML also allows additional authentication schemes, but does not mandate them. A list of additional types follows (but is not the definitive list):

Authentication schemes	Description
syncml:auth-x509	The data would be an actual X.509 Certificate. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.

syncml:auth-securid	The data specific for SecurID authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.
syncml:auth-safeword	The data specific for SafeWord authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.
syncml:auth-digipass	The data specific for DigiPass authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.

Note that if a challenge from the authentication server is to be sent, it will be sent in the `NextNonce` element.

5.4 XML Usage

The SyncML Messages are represented in a mark-up language defined by [XML]. The SyncML representation protocol is an XML application. The SyncML DTD (Document Type Definition) defines the XML document type used to represent a SyncML Message. The SyncML DTD can be found in Section 7, but it is not necessary to read the DTD in order to understand the protocol.

SyncML Messages are specified using well-formed XML. However, the SyncML Messages need not be valid XML. That is, the SyncML Messages do not need to specify the XML declaration or prolog. They only need to specify the body of the XML document. This restriction allows for the SyncML Messages to be specified with greater terseness than well-formed, valid XML documents.

SyncML makes heavy use of XML name spaces. Name spaces MUST be declared on the first element type that uses an element type from the name space.

Names in XML are case sensitive. By convention in the SyncML DTD, the element type and attribute list names are specified using the convention that the first character in each word of the name is in upper case text and remainder of the characters in each word of the names specified in lower case text. For example, `SyncML` for the Sync Mark-up Language tag or `MsgRef` for the Message Reference tag.

The element types in the SyncML DTD are defined within a namespace associated with the URI `http://www.syncml.org/docs/syncml_represent_v11_20020215.dtd` or the URN `SYNCML:SYNCML1.1`. The SyncML DTD are also identified by the ISO 9070 formal public identifier `-//SYNCML//DTD SyncML 1.1//EN`.

SyncML also makes use of XML standard attributes, such as `xml:lang`. Any XML standard attribute can be used in a SyncML document.

XML can be viewed as more verbose than alternative binary representations. This is often cited as a reason why it may not be appropriate for low bandwidth network protocols. In most cases, SyncML uses shortened element type and attribute names. This provides a minor reduction in verbosity. Additionally, the SyncML Messages can be encoded in a tokenized, binary format defined by [WBXML]. The use of [WBXML] format is external to specification of the SyncML protocol and should be transparent to any SyncML application. The combination of the use of shortened element type names and an alternative binary format makes SyncML competitive, from a compressed format perspective, with alternative, but private, binary representations.

One of the main advantages of XML is that it is a widely accepted International recommendation for text document mark-up. It provides for both human readability and machine processability. In addition, XML allows the originator to capture the

structure of a document, not just its content. This is extremely useful for applications such as data synchronization, where not just content, but structure semantics is often exchanged.

5.5 MIME Usage

The [RFC2045] Internet standard provides an industry-accepted mechanism for identifying different content types. The SyncML Message is identified by a MIME media type. The media type for the SyncML Message is registered within the vendor tree. The MIME content types for SyncML Data Synchronization are specified in [DSREPU]. The MIME content types for SyncML Device Management are specified in [DMREPU]. One of these MIME content types MUST be used for identifying SyncML Messages within transport and session level protocols that support MIME content types.

5.6 Identifiers

Identifiers in SyncML, such as in the `Source` or `Target` element types, can be a combination of Uniform Resource Identifiers (URI), as defined by [RFC2396], Uniform Resource Names (URN) and textual names.

In SyncML, all URI and URN values are specified as parsable character data in element types or as character data in attribute lists. Applications MUST specify a valid URI or URN value. Even with an integrated "validating XML parser", as defined in [XML], an application will need to confirm the validity of any URI or URN.

SyncML uses the SYNCML URN type to identify SyncML specific name spaces and unique names. Other URN types may be used. For instance, the `LocURI` element type could contain one of the following URN:

IMEI URN	Identify an International Mobile Equipment Identifiers [IMEI] . The IMEI URN specifies a valid, 15 digit IMEI. The format of the URN is IMEI: ###############
ESN URN	Identify an Electronic Serial Number. The ESN specifies a valid, 8 digit ESN. The format of the URN is ESN: ########
MEID URN	Identify an Mobile Equipment Identity. The MEID URN specifies a valid, 15 digit MEID. The format of the URN is MEID: ###############

Other URN types may be used in the `LocURI` element type also.

6. Mark-up Language Description

The SyncML representation protocol is a document mark-up consisting of XML element types. This section provides a prose description of this mark-up. The element types are defined in terms of their purpose or usage, parent elements, any restrictions on content or use and content model.

In the "Attributes" section for the description for each element type, the text "None." means no SyncML-specific attributes are defined. However, XML standard attributes can still be used within these element types.

Restrictions listed in this document apply to all SyncML protocols. Any additional restrictions for each protocol and examples that illustrate the use of each element type can be found in the representation protocol usage documents [DSREPU] and [DMREPU].

6.1 Common Use Elements

The following are common element types used by numerous other SyncML element types.

6.1.1 Archive

Usage: Indicator that the data specified in the `Delete` command SHOULD be archived (e.g., a copy kept) by the recipient prior to being deleted from the recipient data collection.

Parent Elements: `Delete`

Content Model:

```
(EMPTY)
```

Attributes: None.

6.1.2 Chal

Usage: Specifies an authentication challenge. The receiver of the challenge specifies authentication credentials, of the given authentication type and format, in the next request.

Parent Elements: `Status`

Restrictions: The `Meta` element type specifies any meta-information about the challenge. The `Type` and `Format` element types within the `Meta` element type specify the authentication scheme type and format, respectively. The default type is `syncml:auth-basic` for the SyncML "Basic" form of authentication. The type value `syncml:auth-md5` MUST be explicitly specified to indicate the SyncML "MD5 Digest Access" authentication scheme. If the SyncML "MD5 Digest Access" authentication scheme is used, the `NextNonce` element type can be specified if the challenger requests the use of a new nonce string. The format value MUST be `b64`, when using the clear-text, XML representation. The types for the SyncML authentication schemes are specified in Section 4.3, "Security", of this specification.

An authentication challenge can be specified for each of a number of SyncML "security layers". For example, a challenge can be specified against the SyncML server, database or an individual command on a database. To challenge a SyncML server, a `Chal` element type is sent in the `Status` command corresponding to the `SyncHdr` of the associated SyncML request. To challenge a database, the `Chal` element type is sent in the `Status` command corresponding to the `Alert` or `Sync` command associated with the database. To challenge a command on a database, the `Chal` element type is sent in the `Status` command corresponding to an individual command (e.g., `Add`, `Alert`, `Delete`) on the database. Mechanisms for authentication challenges at the transport level are handled within the individual transport.

If absent and if the status code is (`200`) `OK`, then the same credentials MUST be used in the next SyncML request.

If absent and if the status code is (`212`) `Authentication accepted`, then credentials need not be specified for any subsequent SyncML requests within the current session. The session is authenticated.

Content Model:

```
(Meta)
```

Attributes: None.

6.1.3 Cmd

Usage: Specifies the name of the SyncML command referenced by a `Status` element type.

Parent Elements: `Status`

Restrictions: The value MUST be one of `Add`, `Alert`, `Atomic`, `Copy`, `Delete`, `Exec`, `Get`, `Map`, `Put`, `Replace`, `Results`, `Search`, `Sequence`, `Status`, `Sync`.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.4 CmdID

Usage: Specifies a SyncML message-unique command identifier.

Parent Elements: `Add`, `Alert`, `Atomic`, `Copy`, `Delete`, `Exec`, `Get`, `Map`, `Put`, `Replace`, `Results`, `Search`, `Sequence`, `Status`, `Sync`

Restrictions: A text value that MUST be unique within the SyncML Message.

The element type MUST always be present and the value MUST NOT be the text string "0".

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.5 CmdRef

Usage: Specifies the `CmdID` referenced by a `Status` element type.

Parent Elements: `Results`, `Status`

Restrictions: MUST refer to the identifier of the SyncML command reference by the `Status` element type.

The only instance where the element type can be absent in the `Status` command is the case where the `Status` command refers to the `SyncHdr` of the associated SyncML request message. For example, a status can be sent back to the originator for exceptions (e.g., (`401`) `Unauthorized`) found within the `SyncHdr` of the originator's request.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.6 Cred

Usage: Specifies an authentication credential for the originator.

Parent Elements: Add, Alert, Copy, Delete, Exec, Get, Put, Map, Replace, Search, Status, Sync, SyncHdr

Restrictions: The Meta element type specifies any meta-information about the credentials. The Type and Format element types within the Meta element type specify the credential scheme type and format, respectively. The default type is syncml:auth-basic for the "Basic" form of authentication. The type value syncml:auth-md5 MUST be explicitly specified to indicate the SyncML "MD5 Digest" authentication scheme. The format MUST be b64, when using the clear-text, XML representation. However, when using "Basic" form of authentication, the b64 format does not indicate that the credentials are base64 encoded twice. The Data element type specifies the credential value. The types for these SyncML authentication schemes are specified in Section 4.3, "Security", of this specification.

If absent, and no other authentication credential was specified in either a parent command or in the SyncHdr element type, then no authentication credential is specified.

If an authentication credential was specified by a parent command or in the SyncHdr element type, then that authentication credential specified there is assumed to be sufficient for the operation specified by the current element type. Specifying insufficient authentication credentials will result in a (401) Unauthorized exception condition.

If the authentication challenge is received (See the Chal element type) for the request, the credential type and format of the next request MUST be applied to it.

Content Model:

```
(Meta?, Data)
```

Attributes: None.

6.1.7 Final

Usage: Indicator that the SyncML message is the last message in the current SyncML package.

Parent Elements: SyncBody

Restrictions: The element type MUST only be specified on the last message of the SyncML package. If not present, then more messages follow this SyncML message in the current SyncML package.

The SyncML Synchronization Protocol specification [DSPRO] specifies the semantics of the different SyncML packages for the SyncML synchronization protocol. The SyncML Device Management Protocol specification [DMPRO] specifies the semantics of the different SyncML packages for the SyncML device management protocol.

Content Model:

```
(EMPTY)
```

Attributes: None.

6.1.8 Lang

Usage: Specifies a preferred language for results data on a Get, Put or Search command (i.e., commands that return text results).

Parent Elements: Get, Put, Search

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.9 LocName

Usage: Specifies the display name for the target or source address.

Parent Elements: `Target`, `Source`

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.10 LocURI

Usage: Specifies the target or source specific address.

Parent Elements: `Target`, `Source`

Restrictions: MUST be either an absolute or a relative URI or a well-known URN.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.11 MoreData

Usage: Indicator that a SyncML data element is incomplete and there will be one or more subsequent chunks.

Parent Elements: `Item`

Restrictions: The element type MUST be specified on all but the last chunk of data of an item. If not present, then the item is either contained within a single message or is the closing chunk of the data item.

Content Model:

```
(EMPTY)
```

Attributes: None.

6.1.12 MsgID

Usage: Specifies a SyncML session-unique identifier for the SyncML Message.

Parent Elements: `SyncHdr`

Restrictions: The message identifier MUST be unique to the device within the SyncML session. The element type MUST be specified in the `SyncHdr`. The value is a monotonically increasing numeric value starting at one (1) for the first message in the SyncML session. The message identifier specified in a SyncML request MUST be the content of the `MsgRef` element type in the corresponding SyncML results or response status.

Content Model:

```
(#PCDATA)
```

Attributes: None

6.1.13 MsgRef

Usage: Specifies a reference to a SyncML session-unique identifier referenced by a SyncML results or response status.

Parent Elements: `Results`, `Status`

Restrictions: The value MUST reference the message identifier of the SyncML message referred to by the results or response status.

Content Model:

```
(#PCDATA)
```

Attributes: None

6.1.14 NoResp

Usage: Indicates that the originator does not want a response status sent back in the response message.

Parent Elements: `Add`, `Alert`, `Atomic`, `Copy`, `Delete`, `Exec`, `Get`, `Put`, `Replace`, `Search`, `Sequence`, `Sync`, and `SyncHdr`

Content Model:

```
(EMPTY)
```

Attributes: None

6.1.15 NoResults

Usage: Indicates that the results of a `Search` command MUST be retained on the recipient to be used as the source for a subsequent SyncML command.

Parent Elements: `Search`

Content Model:

```
(EMPTY)
```

Attributes: None

6.1.16 NumberOfChanges

Usage: Indicates the total number of changes (the number of `Add`, `Replace` and `Delete` commands) that are going to be sent from sender to recipient during a synchronization session so that the recipient may use this information to calculate progress information.

Parent Elements: `Sync`

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.17 RespURI

Usage: Specifies the URI that the recipient MUST use for any response to this message.

Parent Elements: `SyncHdr`

Restrictions: The value of this element is the address, in the form of an absolute URI that the recipient MUST use for any response to this message. If the Source is not the same as this value, then the Source element MUST also be specified in the SyncHdr element type. Note that the server and databases are the same entities at this new address. Receipt of this command does not mean you should repeat commands in the previous message.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.18 SessionID

Usage: Specifies the identifier of the SyncML session associated with the SyncML Message.

Parent Elements: `SyncHdr`

Restrictions: The value is an opaque string. The element type MUST be specified in the SyncHdr element type in all SyncML Messages. The initiator SHOULD use unique SessionIDs for each session.

The maximum length of a `SessionID` is 4 bytes. Note for a client having an 8 bit incrementing `SessionID` counter is enough for practical implementations.

Content Model:

```
(#PCDATA)
```

Attributes: None

6.1.19 SftDel

Usage: Indicates that the delete command is a "Soft Delete".

Parent Elements: `Delete`

Content Model:

```
EMPTY
```

Attributes: None

6.1.20 Source

Usage: Specifies source routing or mapping information.

Parent Elements: `Item, Map`, `MapItem`, `Search`, `Sync`, `SyncHdr`,

Restrictions: When specified in the `Item` element type, the `Source` element type specifies the database item that is the source of the SyncML command.

When specified in the `SyncHdr` element type, the `Source` element type specifies the source routing information for the network device that originated the SyncML Message.

If the `RespURI` element type is also specified within the `SyncHdr`, then the `Source` element type specifies the source routing information for a proxy originator of the SyncML message.

Content Model:

```
(LocURI, LocName?)
```

Attributes: None.

6.1.21 SourceRef

Usage: Specifies the `Source` referenced by a `Status` or `Results` element type

Parent Elements: `Status, Results`

Restrictions: When specified in the `Status` element type, specifies the source address specified in the command associated with the response status. When specified in the `Results` element type, specifies the source address specified in the associated `Search` command.

The element type MUST be specified in a `Status` command corresponding to any SyncML command that includes the `Source` element type.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.22 Target

Usage: Specifies target routing or mapping information.

Parent Elements: `Item, Map`, `MapItem`, `Search`, `Sync`, `SyncHdr`,

Restrictions: When specified in the `Item` element type, the `Target` element type specifies the database item that is the target of the SyncML command.

When specified in the `SyncHdr` element type, the `Target` element type specifies the target routing information for the network device that is receiving the SyncML Message.

Content Model:

```
(LocURI, LocName?)
```

Attributes: None.

6.1.23 TargetRef

Usage: Specifies the `Target` referenced by a `Status` or `Results` element type

Parent Elements: `Status, Results`

Restrictions: When specified in the `Status` element type, specifies the target address specified in the command associated with the response status. When specified in the `Results` element type, specifies the target address specified in the associated `Search` command.

The element type MUST be specified in a `Status` command corresponding to any SyncML command that includes the `Target` element type.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.24 VerDTD

Usage: Specifies the major and minor version identifier of the SyncML representation protocol specification used to represent the SyncML message.

Parent Elements: `SyncHdr`

Restrictions: Major revisions of the specification create incompatible changes that will generally require a new SyncML parser. Minor revisions involve changes that do not impact basic compatibility of the parser. When the XML document conforms to this revision of the SyncML representation protocol specification the value MUST be `1.1`. The element type MUST be included in the SyncHdr.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.1.25 VerProto

Usage: Specifies the version identifier of the SyncML protocol specification used with the SyncML Message.

Parent Elements: `SyncHdr`

Restrictions: The first SyncML Message in each SyncML Package sent from an originator to a recipient MUST include the `VerProto` element type in the `SyncHdr`.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.2 Message Container Elements

The following element types provide the basic container support for the SyncML message.

6.2.1 SyncML

Usage: Specifies the container for a SyncML Message.

Parent Elements: None. This is the root or document element.

Content Model:

```
(SyncHdr, SyncBody)
```

Attributes:

Name	Type	Occurrence	Description
xmlns	CDATA	REQUIRED	Value MUST be the text: 'SYNCML:SYNCML1.1'

6.2.2 SyncHdr

Usage: Specifies the container for the revisioning, routing information in the SyncML message.

Parent Elements: `SyncML`

Restrictions: The optional `Meta` is used to convey meta-information about the SyncML messages, such as the maximum byte size of a SyncML response.

Content Model:

```
(VerDTD, VerProto, SessionID, MsgID, Target, Source, RespURI?, NoResp?, Cred?, Meta?)
```

Attributes: None

6.2.3 SyncBody

Usage: Specifies the container for the body or contents of the SyncML message.

Parent Elements: `SyncML`

Restrictions: None.

Content Model:

```
((Alert | Atomic | Copy | Exec | Get | Map | Put | Results | Search | Sequence | Status | Sync | Add |
Replace | Delete)+, Final?)
```

Attributes: None.

6.3 Data Description Elements

The following element types are used as container elements for data exchanged in a SyncML Message.

6.3.1 Data

Usage: Specifies discrete SyncML data.

Parent Elements: `Alert`, `Cred`, `Item`, `Status`, `Search`

Restrictions: The content information can be either parsable character data or mark-up data. If the element type contains any mark-up, then the name space for the element types MUST be declared on the element types in the content information.

When specified in an `Alert`, the element type specifies the type of alert.

When specified in a `Cred`, the element type specifies the authentication credentials.

When specified in an `Item`, the element type specifies the item data.

When specified in a `Status`, the element type specifies the request status code type.

Content Model:

```
(#PCDATA)
```

Attributes: None.

6.3.2 Item

Usage: Specifies a container for item data.

Parent Elements: `Add`, `Alert`, `Copy`, `Delete`, `Exec`, `Get`, `Put`, `Replace`, `Results`, `Status`

Restrictions: If the source URI for the data is an external entity, then the `Data` element is absent. In this case, the recipient will need to retrieve the data from the specified network location.

Any Data element in Item MUST be the last element in Item.

The LocURI element type in the Target or Source element types for any of the SyncML commands can be a relative URL. This restriction is not captured by the SyncML DTD.

When specified in an Add, Copy, Delete, Exec, Get, Put, Replace, or Results command, the element type specifies data item that is the operand for the command.

When specified in an Alert, the element type specifies the parameters for the alert type.

When specified in a Status, the element type specifies additional information about the request status code type. For example, it might specify the component of the request that caused the status condition.

Content Model:

```
(Target?, Source?, Meta?, Data?)
```

Attributes: None.

6.3.3 Meta

Usage: Specifies meta-information about the parent element type.

Parent Elements: Add, Atomic, Chal, Copy, Cred, Delete, Get, Item, Map, Put, Replace, Results, Search, Sequence, Sync

Restrictions: When specified in the Chal, the element type specifies meta-information about the authentication scheme requested.

When specified in the Cred, the element type specifies meta-information about the authentication credential.

When specified in the Atomic, Sequence and Sync, then the scope for the meta-information includes all the contained commands, unless the meta-information is overridden by a Meta in a contained command.

When specified in the Results, the element type specifies meta-information about the results set.

When specified in the Add, Copy, Delete, Get, and Replace commands, the element type specifies meta-information about the SyncML command.

Content Model:

```
(#PCDATA)
```

Attributes: None

6.4 Protocol Management Elements

6.4.1 Status

Usage: Specifies the request status code for a corresponding SyncML command.

Parent Elements: SyncBody

Restrictions: A Status command only applies to the command corresponding to the specified CmdRef (i.e., 1:1 correspondence of a command and a Status). If there were multiple Item elements specified in the command, and if the items' status code were not the same, then a Status MUST be returned for each of the items. If all of the items had the same status code, a Status for all of the items MAY be returned. In these cases the SourceRef and TargetRef elements are used to identify the Item, which the status code applies to. If all of the items in the command had the same

status code, then it is also allowed to return a single Status for the entire command. When returning a single Status for a command with multiple items, the SourceRef and TargetRef elements MUST NOT be specified in the Status command.

Additionally, if the Status command is associated with a command that had other commands inside it (e.g., Sync, Atomic, Sequence), then the status value only applies to the corresponding command, and is not related to the status of the commands inside it.

Ordering of Status commands in a SyncML response MUST match the order of the commands in the corresponding SyncML request. That is, when there are multiple commands in a SyncML request, then the corresponding Status commands MUST appear in the SyncML response in the same order as the associated commands appeared in the SyncML request.

In addition, the status on the SyncHdr MUST be the first status element in the SyncBody of the response. Even in the case where the statuses for the previous request span multiple messages/responses, the status on SyncHdr MUST be the first status element followed by other statuses and/or remaining statuses.

The CmdID element type specifies the SyncML message-unique identifier for this command.

The MsgRef element type specifies the MsgID of the associated SyncML request.

The CmdRef element type specifies the CmdID of the associated SyncML request. The element type MUST be present. If "0", the Status command corresponds to a status code for the SyncHdr of the SyncML message referenced by the Status command.

The Cmd element type specifies the name of the SyncML command associated with the SyncML request. The value of the element type can also be "SyncHdr" when the CmdRef element type has a value of "0".

The optional TargetRef element type specifies the target addresses from the associated command. If the Item elements of the command associated with the Status command has a Target element, the value MUST be copied into the TargetRef of the Status command. If more than one TargetRef element type is specified, then the request status code applies to all of these TargetRef values. If the request status code is applicable to the entire list of items specified in the associated request command, then the TargetRef element type MUST NOT be specified.

The optional SourceRef element type specifies the source address from the associated command. If the Item elements of the command associated with the Status command has a Source element, the value MUST be copied into the SourceRef of the Status command. If more than one SourceRef element type is specified, then the request status code applies to all of these SourceRef values. If the request status code is applicable to the entire list of items specified in the associated request command, then the SourceRef element type MUST NOT be specified.

The Cred element type specifies authentication credential for the command.

The Chal element type specifies the authentication challenge for the command or the message. If the status code in the Data element is (401) Unauthorized or (407) Authentication required, the challenge SHOULD be included.

The Data element type specifies the request status code type.

The optional and repeatable Item element type contains additional information about the status condition, such as the SyncML command.

This specification permits a Status command to be issued against another Status command. This case will probably not normally be encountered. However, there are extreme cases where this feature is necessary. For example, if a server returns a (401) Unauthorized status code with a request for an authentication scheme that is not supported by the client, the client might use a (406) Optional feature unsupported to notify the server that that requested authentication scheme is not supported and negotiate a authentication scheme it does support. SyncML servers and SyncML clients not supporting such a usage case need provide no further response to the SyncML entity issuing the "Status on a Status".

A `Status` MUST also be returned for the `SyncHdr`. However, if a client creates a message containing only a successful `Status` on a `SyncHdr`, the entire message MUST NOT be sent. A server MUST send this message.

Status codes are listed in Section 11, Response Status Codes.

Content Model:

(CmdID, MsgRef, CmdRef, Cmd, TargetRef*, SourceRef*, Cred?, Chal?, Data, Item*)

Attributes: None.

6.5 Protocol Command Elements

6.5.1 Add

Usage: Specifies the SyncML command to add data to a data collection.

Parent Elements: `Atomic`, `Sequence`, `Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Cred?, Meta?, Item+)

Attributes: None.

6.5.2 Alert

Usage: Specifies the SyncML command for sending custom content information to the recipient. The command provides a mechanism for communicating content information, such as state information or notifications to an application on the recipient device. In addition, this command provides a "standard way to specify non-standard" extended commands, beyond those defined in this specification.

Parent Elements: `Atomic`, `Sequence`, `SyncBody`

Content Model:

(CmdID, NoResp?, Cred?, Data?, Item*)

Attributes: None.

6.5.3 Atomic

Usage: Specifies the SyncML command to request that the subordinate commands be executed as a set or not at all.

Parent Elements: `Sequence`, `Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Meta?, (Add | Delete | Copy | Atomic | Map | Replace | Sequence | Sync | `Get` | `Exec` | `Alert`)+)

Attributes: None.

6.5.4 Copy

Usage: Specifies that the SyncML command to copy data items from one location to another in the recipient's database.

Parent Elements: `Atomic`, `Sequence`, `Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Cred?, Meta?, Item+)

Attributes: None.

6.5.5 Delete

Usage: Specifies the SyncML command to delete data from a data collection.

Parent Elements: `Atomic`, `Sequence`, `Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Archive?, SftDel?, Cred?, Meta?, Item+)

Attributes: None.

6.5.6 Exec

Usage: Specifies the SyncML command to execute a process on the recipient network device.

Parent Elements: `SyncBody`, `Atomic`, `Sequence`

Content Model:

(CmdID, NoResp?, Cred?, Meta?, Item)

Attributes: None.

6.5.7 Get

Usage: Specifies the SyncML command to retrieve data from the recipient.

Parent Elements: `SyncBody`, `Sequence`, `Atomic`

Content Model:

(CmdID, NoResp?, Lang?, Cred?, Meta?, Item+)

Attributes: None.

6.5.8 Map

Usage: Specifies the SyncML command used to update identifier maps.

Parent Elements: `Atomic`, `Sequence`, `SyncBody`

Content Model:

(CmdID, Target, Source, Cred?, Meta?, MapItem+)

Attributes: None.

6.5.9 MapItem

Usage:

Parent Elements: `Map`

Content Model:

(Target, Source)

Attributes: None.

6.5.10 Put

Usage: Specifies the SyncML command to transfer data items to a recipient network device or database.

Parent Elements: `SyncBody`

Content Model:

(CmdID, NoResp?, Lang?, Cred?, Meta?, Item+)

Attributes: None.

6.5.11 Replace

Usage: Specifies the SyncML command to replace data.

Parent Elements: `Atomic, Sequence, Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Cred?, Meta?, Item+)

Attributes: None.

6.5.12 Results

Usage: Specifies the SyncML command that is used to return the results of a `Search` or `Get` command.

Parent Elements: `SyncBody`

Restrictions: The mandatory `CmdID` element type specifies the SyncML message-unique identifier for this command.

The optional `MsgRef` element type specifies the MsgID of the associated SyncML request. If the `MsgRef` is not present in a `Results` element type, then the `MsgRef` value of "1" MUST be assumed.

The `CmdRef` element type specifies the `CmdID` of the associated SyncML request. If not present, the response status code is associated with `CmdID` value of "1".

The optional `Meta` element type specifies meta-information to be used for the command. For example, the common media type or format for all the items can be specified. The scope of the meta-information is limited to the command.

The optional `TargetRef` element type specifies the target address from the associated command.

The optional `SourceRef` element type specifies the source address from the associated command.

One or more `Item` element types MUST be specified. The `Item` element type specifies the results. The `Source` specified within the `Item` element type SHOULD be a relative URI, as relative to the corresponding `SourceRef`.

Content Model:

(CmdID, MsgRef?, CmdRef, Meta?, TargetRef?, SourceRef?, Item+)

Attributes: None

6.5.13 Search

Usage: Specifies the SyncML command to search a recipient database.

Parent Elements: `SyncBody`

Content Model:

(CmdID, NoResp?, NoResults?, Cred?, Target?, Source+, Lang?, Meta, Data)

Attributes: None.

6.5.14 Sequence

Usage: Specifies the SyncML command to order the processing of a set of SyncML commands.

Parent Elements: `Atomic`, `Sync`, `SyncBody`

Content Model:

(CmdID, NoResp?, Meta?, (Add | Replace | Delete | Copy | Atomic | Map | Sync | Get | Alert | Exec)+)

Attributes: None.

6.5.15 Sync

Usage: Specifies the SyncML command that indicates a data synchronization operation.

Parent Elements: `Atomic`, `Sequence`, `SyncBody`

Content Model:

(CmdID, NoResp?, Cred?, Target?, Source?, Meta?, (Add | Atomic | Copy | Delete | Sequence | Replace)*)

Attributes: None.

7. SyncML DTD

```
<!--

SyncML Representation Protocol (SYNCML) V1.1 Document Type Definition


Copyright Open Mobile Alliance Ltd., 2002-2003

          All rights reserved


SYNCML is an XML language. Typical usage:

   <?xml version="1.0"?>

   <!DOCTYPE SyncML PUBLIC "-//OMA//DTD SYNCML 1.1//EN"

             "http://www.openmobilealliance.org/DTD/OMA-SyncML-
Representation_Protocol-DTD-1.1.2-20030221-D.dtd"

             [<?oma-syncml-ver supported-versions="1.1"?>]>

   <SyncML>

      ...

   </SyncML>


Terms and conditions of use are available from the

Open Mobile Alliance Ltd. web site at

http://www.openmobilealliance.org/useterms.html

-->


<?xml version="1.0" encoding="UTF-8"?>

<!-- Root Element -->

<!ELEMENT SyncML (SyncHdr, SyncBody)>

<!ELEMENT SyncHdr (VerDTD, VerProto, SessionID, MsgID, Target, Source,
RespURI?, NoResp?, Cred?, Meta?)>

<!ELEMENT SyncBody ((Alert | Atomic | Copy | Exec | Get | Map | Put |
Results | Search | Sequence | Status | Sync | Add | Replace | Delete)+,
Final?)>

<!-- Commonly Used Elements -->

<!-- Archive indicator for Delete -->

<!ELEMENT Archive EMPTY>

<!—Value must be one of "Add" | "Alert" | "Atomic" | "Copy" | "Delete" |
```

```
"Exec" | "Get" | "Map" | "Put" | "Replace" | "Results" | "Search" |
"Sequence" | "Status" | "Sync". -->

<!ELEMENT Cmd (#PCDATA)>

<!-- Authentication Challenge -->

<!ELEMENT Chal (Meta)>

<!-- Sync message unique identifier for command -->

<!ELEMENT CmdID (#PCDATA)>

<!-- Reference to command identifier -->

<!ELEMENT CmdRef (#PCDATA)>

<!-- Credentials -->

<!ELEMENT Cred (Meta?, Data)>

<!-- Final message flag -->

<!ELEMENT Final EMPTY>

<!-- Desired language for results -->

<!ELEMENT Lang (#PCDATA)>

<!-- Location displayable name -->

<!ELEMENT LocName (#PCDATA)>

<!-- Location URI -->

<!ELEMENT LocURI (#PCDATA)>

<!-- Indication for more data to come -->

<!ELEMENT MoreData EMPTY>

<!-- SyncML Message ID -->

<!ELEMENT MsgID (#PCDATA)>

<!-- Reference to a SyncML Message ID -->

<!ELEMENT MsgRef (#PCDATA)>

<!-- No Response Status Requested Indicator -->

<!ELEMENT NoResp EMPTY>

<!-- No Results Requested Indicator -->

<!ELEMENT NoResults EMPTY>

<!-- NumberOfChanges used to display progress information -->

<!ELEMENT NumberOfChanges (#PCDATA)>

<!-- URI recipient used for response -->
```

```
<!ELEMENT RespURI (#PCDATA)>

<!-- SyncML session identifier -->

<!ELEMENT SessionID (#PCDATA)>

<!-- Soft delete indicator for Delete -->

<!ELEMENT SftDel EMPTY>

<!-- Source location -->

<!ELEMENT Source (LocURI, LocName?)>

<!ELEMENT SourceRef (#PCDATA)>

<!-- Target location information -->

<!ELEMENT Target (LocURI, LocName?)>

<!ELEMENT TargetRef (#PCDATA)>

<!-- SyncML specification major/minor version info. -->

<!-- For this version of the DTD, the value is "1.1" -->

<!ELEMENT VerDTD (#PCDATA)>

<!-- Data sync protocol major/minor version -->

<!-- For example, "xyz/1.1" -->

<!ELEMENT VerProto (#PCDATA)>

<!-- Synchronization data elements -->

<!-- Item element type -->

<!ELEMENT Item (Target?, Source?, Meta?, Data?, MoreData?)>

<!-- Meta element type -->

<!-- Element types in the content MUST have name space declared. -->

<!--The Meta content would be something such as: <Meta> <Type
xmlns='syncml:metinf'>text/calendar</Type> <Format
xmlns='syncml:metinf'>xml</Format> </Meta>-->

<!ELEMENT Meta (#PCDATA)>

<!-- Actual data content -->

<!ELEMENT Data (#PCDATA)>

<!-- SyncML Commands -->

<!-- Add operation. -->

<!ELEMENT Add (CmdID, NoResp?, Cred?, Meta?, Item+)>

<!-- Alert operation. -->

<!-- Alert types are either "User Agent" or "Application" oriented -->
```

```
<!ELEMENT Alert (CmdID, NoResp?, Cred?, Data?, Item*)>

<!-- Atomic operation. All or nothing semantics. -->

<!ELEMENT Atomic (CmdID, NoResp?, Meta?, (Add | Replace | Delete | Copy |
Atomic | Map | Sequence | Sync | Get | Exec | Alert)+)>

<!-- Copy operation. -->

<!ELEMENT Copy (CmdID, NoResp?, Cred?, Meta?, Item+)>

<!-- Delete operation. -->

<!ELEMENT Delete (CmdID, NoResp?, Archive?, SftDel?, Cred?, Meta?, Item+)>

<!-- Exec operation -->

<!-- Executable can either be referenced with Target element type -->

<!-- or can be specified in the Data element type. -->

<!ELEMENT Exec (CmdID, NoResp?, Cred?, Meta?, Item)>

<!-- Get operation. -->

<!ELEMENT Get (CmdID, NoResp?, Lang?, Cred?, Meta?, Item+)>

<!-- MAP operation. Create/Delete an item id map kept at the server. -->

<!ELEMENT Map (CmdID, Target, Source, Cred?, Meta?, MapItem+)>

<!ELEMENT MapItem (Target, Source)>

<!-- Put operation. -->

<!ELEMENT Put (CmdID, NoResp?, Lang?, Cred?, Meta?, Item+)>

<!-- Replace operation. -->

<!ELEMENT Replace (CmdID, NoResp?, Cred?, Meta?, Item+)>

<!-- Results operation. -->

<!ELEMENT Results (CmdID, MsgRef?, CmdRef, Meta?, TargetRef?, SourceRef?,
Item+)>

<!-- Search operation. -->

<!ELEMENT Search (CmdID, NoResp?, NoResults?, Cred?, Target?, Source+,
Lang?, Meta, Data)>

<!-- Sequence operation. -->

<!ELEMENT Sequence (CmdID, NoResp?, Meta?, (Add | Replace | Delete | Copy
| Atomic | Map | Sync | Get | Alert | Exec)+)>

<!-- Status operation. -->

<!ELEMENT Status (CmdID, MsgRef, CmdRef, Cmd, TargetRef*, SourceRef*,
Cred?, Chal?, Data, Item*)>

<!-- Synchronize Operation. -->
```

```
<!ELEMENT Sync (CmdID, NoResp?, Cred?, Target?, Source?, Meta?,
NumberOfChanges?, (Add | Atomic | Copy | Delete | Replace | Sequence)*)>

<!-- End of DTD Definition -->
```

8. WBXML Definition

The following tables define the token assignments for the mapping of the SyncML related DTDs and element types into WBXML as defined by [WBXML].

8.1 Code Space Definitions

This version of the SyncML representation protocol specification maps all the SyncML related DTDs into WBXML code spaces.

DTD Name	WBXML PUBLICID Token (Hex Value)	Formal Public Identifier
SyncML	FD1	-//SYNCML//DTD SyncML 1.0//EN
SyncML 1.1	FD3	-//SYNCML//DTD SyncML 1.1//EN

The SyncML DTD is assigned the WBXML document public identifier (i.e., the "publicid" WBXML BNF production) associated with the FD1 or FD3 token.

8.2 Code Page Definitions

The following code page tokens represent SyncML related DTD public identifiers. This version of the SyncML representation protocol specification utilizes the WBXML code page tokens for identifying DTDs.

DTD Name	WBXML Code Page Token (Hex Value)	Formal Public Identifier
SyncML	00	-//SYNCML//DTD SyncML 1.1//EN
MetInf	01	-//SYNCML//DTD MetInf 1.1//EN

8.3 Token Definitions

The following WBXML token codes represent element types (i.e., tags) form code page x00 (zero), SyncML DTD.

Element Type Name	WBXML Tag Token (Hex Value)
Add	05
Alert	06
Archive	07
Atomic	08
Chal	09
Cmd	0A
CmdID	0B
CmdRef	0C
Copy	0D
Cred	0E

Data	0F
Delete	10
Exec	11
Final	12
Get	13
Item	14
Lang	15
LocName	16
LocURI	17
Map	18
MapItem	19
Meta	1A
MsgID	1B
MsgRef	1C
NoResp	1D
NoResults	1E
Put	1F
Replace	20
RespURI	21
Results	22
Search	23
Sequence	24
SessionID	25
SftDel	26
Source	27
SourceRef	28
Status	29
Sync	2A
SyncBody	2B

SyncHdr	2C
SyncML	2D
Target	2E
TargetRef	2F
Reserved for future use.	30
VerDTD	31
VerProto	32
NumberOfChanges	33
MoreData	34

The WBXML token codes from code page x01 (one) represent the MetInf DTD. These token definitions are defined in the MetInf DTD specification [DMPRO].

9. Common URI Scheme Types

The following is a list of common URI scheme types

URI Scheme Type	Description
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IMEI	International Mobile Equipment Identifier
LDAP	Lightweight Directory Access Protocol
OBEX	IrDA Object Exchange Protocol
SYNCML	SyncML specific, as defined in one of the protocol or format specifications
WSP	Wireless Session Protocol
ESN	Electronic Serial Number
MEID	Mobile Equipment Identity

10.Base Media Types

The following common, base media types that are referenced by this specification. Any other IANA registered media type can be conveyed by SyncML.

Media Type	Name	URI
application/vnd.syncml-devinf+xml	SyncML Device Info v1.1 (clear text xml)	http://www.syncml.org/docs/syncml_devinf_v11_20020213.pdf
application/vnd.syncml-devinf+wbxml	SyncML Device Info v1.1 (wbxml)	http://www.syncml.org/docs/syncml_devinf_v11_20020213.pdf
text/plain	Plain Text	http://www.ietf.org/rfc/rfc2046.txt
text/x-vcard	vCard v2.1	http://imc.org/pdi/vcard-21.doc
text/vcard	vCard v3.0	http://www.ietf.org/rfc/rfc2426.txt
application/vnd.syncml-xcard	XML vCard v3.0	TBD
text/x-vcalendar	vCalendar	http://www.imc.org/pdi/vcal-10.doc
text/calendar	Icalendar	http://www.ietf.org/rfc/rfc2445.txt
applicaton/vnd.syncml-xcal	XML iCalendar	TDB
text/message	MIME Message	http://www.ietf.org/rfc/rfc2045.txt
application/vnd.syncml-xmsg	XML Email	TBD
application/vnd.syncml-xbookmark	XML Bookmark	TBD
application/vnd.syncml-xrelational	SyncML Relational Object	TBD

11.Response Status Codes

The response status codes in SyncML are a numeric text value. The codes are divided into five classes. The only valid values are the standard values defined in this specification.

Implementations that desire to add to these values SHOULD submit a change request to mailto:technical-comments@openmobilealliance.org.

Status Codes	Reason Phrase
Informational 1xx	
101	In progress. The specified SyncML command is being carried out, but has not yet completed.
Successful 2xx	
200	OK. The SyncML command completed successfully.
201	Item added. The requested item was added.
202	Accepted for processing. The request to either run a remote execution of an application or to alert a user or application was successfully performed.
203	Non-authoritative response. The request is being responded to by an entity other than the one targeted. The response is only to be returned when the request would have been resulted in a 200 response code from the authoritative target.
204	No content. The request was successfully completed but no data is being returned. The response code is also returned in response to a Get when the target has no content.
205	Reset content. The source should update their content. The originator of the request is being told that their content should be synchronized to get an up to date version.
206	Partial content. The response indicates that only part of the command was completed. If the remainder of the command can be completed later, then when completed another appropriate completion request status code SHOULD be created.
207	Conflict resolved with merge. The response indicates that the request created a conflict; which was resolved with a merge of the client and server instances of the data. The response includes both the Target and Source URLs in the `Item` of the `Status`. In addition, a Replace command is returned with the merged data.
208	Conflict resolved with client's command "winning". The response indicates that there was an update conflict; which was resolved by the client command winning.
209	Conflict resolved with duplicate. The response indicates that the request created an update conflict; which was resolved with a duplication of the client's data being created in the server database. The response includes both the target URI of the duplicate in the `Item` of the `Status`. In addition, in the case of a two-way synchronization, an `Add` command is returned with the duplicate data definition.
210	Delete without archive. The response indicates that the requested data was successfully deleted, but that it was not archived prior to deletion because this optional feature was not supported by the implementation.

211	`Item` not deleted. The requested item was not found. It may have been previously deleted.
212	Authentication accepted. No further authentication is needed for the remainder of the synchronization session. This response code can only be used in response to a request in which the credentials were provided.
213	Chunked item accepted and buffered.
214	Operation cancelled. The SyncML command completed successfully, but no more commands will be processed within the session.
215	Not executed. A command was not executed, as a result of user interaction and user chose not to accept the choice.
216	Atomic roll back OK. A command was inside `Atomic` element and `Atomic` failed. This command was rolled back successfully.
Redirection 3xx	
300	Multiple choices. The requested target is one of a number of multiple alternatives requested target. The alternative SHOULD also be returned in the `Item` element type in the `Status`.
301	Moved permanently. The requested target has a new URI. The new URI SHOULD also be returned in the `Item` element type in the `Status`.
302	Found. The requested target has temporarily moved to a different URI. The original URI SHOULD continue to be used. The URI of the temporary location SHOULD also be returned in the `Item` element type in the `Status`. The requestor SHOULD confirm the identity and authority of the temporary URI to act on behalf of the original target URI.
303	See other. The requested target can be found at another URI. The other URI SHOULD be returned in the `Item` element type in the `Status`.
304	Not modified. The requested SyncML command was not executed on the target. This is an additional response that can be added to any of the other Redirection response codes.
305	Use proxy. The requested target MUST be accessed through the specified proxy URI. The proxy URI SHOULD also be returned in the `Item` element type in the `Status`.
Originator Exceptions 4xx	
400	Bad request. The requested command could not be performed because of malformed syntax in the command. The malformed command MAY also be returned in the `Item` element type in the `Status`.
401	Invalid credentials. The requested command failed because the requestor MUST provide proper authentication. If the property type of authentication was presented in the original request, then the response code indicates that the requested command has been refused for those credentials.
402	Payment required. The requested command failed because proper payment is required. This version of SyncML does not standardize the payment mechanism.

403	Forbidden. The requested command failed, but the recipient understood the requested command. Authentication will not help and the request SHOULD NOT be repeated. If the recipient wishes to make public why the request was denied, then a description MAY be specified in the `Item` element type in the `Status`. If the recipient does not wish to make public why the request was denied then the response code 404 MAY be used instead.
404	Not found. The requested target was not found. No indication is given as to whether this is a temporary or permanent condition. The response code 410 SHOULD be used when the condition is permanent and the recipient wishes to make this fact public. This response code is also used when the recipient does not want to make public the reason for why a requested command is not allowed or when no other response code is appropriate.
405	Command not allowed. The requested command is not allowed on the target. The recipient SHOULD return the allowed command for the target in the `Item` element type in the `Status`.
406	Optional feature not supported. The requested command failed because an optional feature in the request was not supported. The unsupported feature SHOULD be specified by the `Item` element type in the `Status`.
407	Missing credentials. This response code is similar to 401 except that the response code indicates that the originator MUST first authenticate with the recipient. The recipient SHOULD also return the suitable challenge in the `Chal` element type in the `Status`.
408	Request timeout. An expected message was not received within the required period of time. The request can be repeated at another time. The `RespURI` can be used to specify the URI and optionally the date/time after which the originator can repeat the request. See `RespURI` for details.
409	Conflict. The requested failed because of an update conflict between the client and server versions of the data. Setting of the conflict resolution policy is outside the scope of this version of SyncML. However, identification of conflict resolution performed, if any, is within the scope.
410	Gone. The requested target is no longer on the recipient and no forwarding URI is known.
411	Size required. The requested command MUST be accompanied by byte size or length information in the `Meta` element type.
412	Incomplete command. The requested command failed on the recipient because it was incomplete or incorrectly formed. The recipient SHOULD specify the portion of the command that was incomplete or incorrect in the `Item` element type in the `Status`.
413	Request entity too large. The recipient is refusing to perform the requested command because the requested item is larger than the recipient is able or willing to process. If the condition is temporary, the recipient SHOULD also include a `Status` with the response code 418 and specify a `RespURI` with the response URI and optionally the date/time that the command SHOULD be repeated.
414	URI too long. The requested command failed because the target URI is too long for what the recipient is able or willing to process. This response code is seldom encountered, but is used when a

	recipient perceives that an intruder may be attempting to exploit security holes or other defects in order to threaten the recipient.
415	Unsupported media type or format. The unsupported content type or format SHOULD also be identified in the `Item` element type in the `Status`.
416	Requested size too big. The request failed because the specified byte size in the request was too big.
417	Retry later. The request failed at this time and the originator should retry the request later. The recipient SHOULD specify a `RespURI` with the response URI and the date/time that the command SHOULD be repeated.
418	Already exists. The requested `Put` or `Add` command failed because the target already exists.
419	Conflict resolved with server data. The response indicates that the client request created a conflict; which was resolved by the server command winning. The normal information in the `Status` should be sufficient for the client to "undo" the resolution, if it is desired.
420	Device full. The response indicates that the recipient has no more storage space for the remaining synchronization data. The response includes the remaining number of data that could not be returned to the originator in the `Item` of the `Status`.
421	Unknown search grammar. The requested command failed on the server because the specified search grammar was not known. The client SHOULD re-specify the search using a known search grammar.
422	Bad CGI Script. The requested command failed on the server because the CGI scripting in the `LocURI` was incorrectly formed. The client SHOULD re-specify the portion of the command that was incorrect in the `Item` element type in the `Status`.
423	Soft-delete conflict. The requested command failed because the "Soft Deleted" item was previously "Hard Deleted" on the server.
424	Size mismatch. The chunked object was received, but the size of the received object did not match the size declared within the first chunk.
425	Permission Denied. The requested command failed because the sender does not have adequate access control permissions (ACL) on the recipient.
Recipient Exception 5xx	
500	Command failed. The recipient encountered an unexpected condition which prevented it from fulfilling the request
501	Command not implemented. The recipient does not support the command required to fulfill the request. This is the appropriate response when the recipient does not recognize the requested command and is not capable of supporting it for any resource.
502	Bad gateway. The recipient, while acting as a gateway or proxy, received an invalid response from the upstream recipient it accessed in attempting to fulfill the request.
503	Service unavailable. The recipient is currently unable to handle the request due to a temporary overloading or maintenance of the recipient. The implication is that this is a temporary condition;

	which will be alleviated after some delay. The recipient SHOULD specify the URI and date/time after which the originator should retry in the `RespURI` in the response.
504	Gateway timeout. The recipient, while acting as a gateway or proxy, did not receive a timely response from the upstream recipient specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary recipient (e.g. DNS) it needed to access in attempting to complete the request.
505	DTD Version not supported. The recipient does not support or refuses to support the specified version of SyncML DTD used in the request SyncML Message. The recipient MUST include the versions it does support in the `Item` element type in the `Status`.
506	Processing error. An application error occurred while processing the request. The originator should retry the request. The `RespURI` can contain the URI and date/time after which the originator can retry the request.
507	`Atomic` failed. The error caused all SyncML commands within an `Atomic` element type to fail.
508	Refresh required. An error occurred that necessitates a refresh of the current synchronization state of the client with the server. Client is requested to initiate a slow sync with the server.
509	Reserved for future use.
510	Data store failure. An error occurred while processing the request. The error is related to a failure in the recipient data store.
511	Server failure. A severe error occurred in the server while processing the request. The originator SHOULD NOT retry the request.
512	Synchronization failed. An application error occurred during the synchronization session. The originator should restart the synchronization session from the beginning.
513	Protocol Version not supported. The recipient does not support or refuses to support the specified version of the SyncML Synchronization Protocol used in the request SyncML Message. The recipient MUST include the versions it does support in the `Item` element type in the `Status`.
514	Operation cancelled. The SyncML command was not completed successfully, since the operation was already cancelled before processing the command. The originator should repeat the command in the next session.
516	Atomic roll back failed. Command was inside `Atomic` element and `Atomic` failed. This command was not rolled back successfully. Server should take action to try to recover client back into original state.
517	Atomic response too large to fit. The response to an atomic command was too large to fit in a single message.

Appendix A. Static Conformance Requirements (Normative)

The static conformance requirements for this specification is specified in [DSREPU], [DSPRO] and [DMCONF].

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
OMA-SyncML-RepPro-V1_1_2-20030612-A	12 Jun 2003	Approved by TP. TP ref# OMA-TP-2003-0265R1
OMA-SyncML-RepPro-V1_1_2-20040711-A	11 Jul 2004	Clerical Changes. Notice sent to TP of minor update TP ref# OMA-TP-2004-0309