SyncML Meta Information

Approved Version 1.2 – 21 Feb 2007

Open Mobile Alliance

OMA-TS-SyncML_MetaInfo-V1_2-20070221-A

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. 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 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 data synchronization specification that contains the following main components:

- An XML-based representation protocol
- A synchronization protocol
- Transport bindings for the synchronization protocol

The data representation specifies an XML DTD that allows the representation of all the information required to perform synchronization, including data, metadata and commands. The synchronization protocols specifies 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.

The synchronization protocol supports both two-way and one-way synchronization.

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, and synchronization 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.

Either the client or the server may initiate a synchronization session, and both one and two-way synchronization are supported. Both linear (point-to-point) and star (one-to-many) synchronization topologies may be implemented using SyncML.

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 outlines the SyncML Meta Information Specification and the respective conformance requirements for clients and servers implementing claiming compliance to it as defined by Open Mobile Alliance across the specification baseline.

2. References

2.1 Normative References

[DSREPU] "SyncML Representation Protocol, Data Synchronization Usage", Open Mobile Alliance™, OMA-TS-DS_DataSyncRep-V1_2, URL:http://www.openmobilealliance.org/

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

[ISO8601] "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601:2000, http://www.iso.ch

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

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[SYNCPRO] "SyncML Synchronization Protocol", Open Mobile Alliance™, OMA-TS-DS_DataSyncProtocol-V1_2, URL:http:www.openmobilealliance.org/

[WBXML] "WAP Binary XML Content Format Specification", WAP Forum,

URL:http://www1.wapforum.org/tech/terms.asp?doc=WAP-192-WBXML-20010725-a.pdf

[XML] "Extensible Markup Language (XML) 1.0", World Wide Web Consortium Recommendation,

http://www.w3.org/TR/REC-xml

[XMLSCHEMADT] "XML Schema Part 2: Datatypes", W3C Recommendation 02 May 2001, http://www.w3.org/XML/Schema

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 {{i1, data1}, ... {in, datan}}}). 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 needed 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.
Meta-Information	Parameter or attributes about the representation, state or type or content of an object or property.
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 and absolutely needed to support SyncML. However, a SyncML implementation that integrates an XML parser might 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.
Request	A message or a command sent from a device to another.
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.
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 Anchor	A string representing a synchronization event. The format of the string will typically be either a sequence number or an ISO 8601-formatted extended representation, basic format date/time stamp.
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.

3.3 Abbreviations

DTD	Document Type Definition
EMI	Experimental Meta Information
OMA	Open Mobile Alliance
URI	Universal Resource Identifier
URN	Universal Resource Name
WBXML	WAP Binary XML
XML	Extensible Markup Language

4. Introduction

The meta-information associated with a SyncML command or data item or collection is represented in a mark-up language defined by [XML]. The meta-information is identifiable as an XML name space. The SyncML Meta-Information DTD (Document Type Definition) defines the XML document type used to represent meta-information used in the [DSREPU] representation protocol. The SyncML Meta-Information DTD can be found in Section 6.

5. Meta Information

5.1 XML Usage

The SyncML Meta-Information XML documents are specified using well-formed XML. However, they need not be valid XML. That is, they do not need to specify the XML prolog. They only need to specify properly identified name space element types from the SyncML Meta-Information DTD. This restriction allows for the SyncML Meta-Information to be specified with greater terseness than would be possible if a well-formed, valid XML document was REQUIRED.

This DTD 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. Element types from the SyncML Meta-Information DTD can be used in other XML documents, including a SyncML message.

Names in XML are case sensitive. By convention in the SyncML Meta-Information DTD, the element type and attribute list names are specified with a "Hungarian" like notation of the first character in each word of the name in upper case text and remainder of the characters in each word of the names specified in lower case text. For example, `MetInf` for the Sync meta-information root element type or `Type` for the content type tag.

The element types in the SyncML Meta-Information DTD are defined within a namespace associated with the URI `http://www.openmobilealliance.org/tech/DTD/OMA-TS-SyncML_MetaInfo_DTD-1_2.dtd` or the URN `syncml:metinf`.

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

XML can be viewed as more verbose than alternative binary representations. This is often cited as a reason why it might not be appropriate for low bandwidth network protocols. In most cases, this DTD uses shortened element type and attribute names. This provides a minor reduction in verbosity. Additionally, the SyncML Meta-Information can be encoded in a tokenized, binary format defined by [WBXML]. The use of [WBXML] format is external to specification of the DTD and transparent to any XML application

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 it's content. This is extremely useful for applications such as data synchronization, where not just content, but structure semantics is often exchanged.

5.2 Element Type Descriptions

5.2.1 Anchor

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current synchronization session.

Parent Elements: `MetInf`

Restrictions: The OPTIONAL Last element type specifies the synchronization anchor for the previous synchronization session. The REQUIRED Next element type specifies the synchronization anchor for the current synchronization session.

Content Model:

```
(Last?, Next)
```

Attributes: None.

Example:

```
<Anchor xmlns='syncml:metinf'>
  <Last xmlns='syncml:metinf'>20000824T133000Z</Last>
  <Next xmlns='syncml:metinf'>20000824T221300Z</Next>
</Anchor>
```

5.2.2 EMI

Usage: Specifies the non-standard, experimental meta information (EMI) extensions supported by the device. The extensions are specified in terms of the XML element type name and the value.

Parent Elements: `MetInf`

Restrictions: The EMI element type MUST specify the extension element name. It MAY also specify one or more enumerated values. Multiple non-standard extensions can be specified by specifying the EMI element type multiple times. This element type is OPTIONAL.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example specifies a non-standard extension with a value of "Confidential".

```
<EMI xmlns='syncml:metinf'>Confidential</EMI>
```

5.2.3 FieldLevel

Usage: Indicates that the content information in the Data element replaces only part of an item.

Parent Elements: `MetInf`

Restrictions: The <FieldLevel/> element MUST only be used if the parent of the <Meta> element is a <Replace> element.

Content Model:

```
EMPTY
```

Attributes: None.

Example:

```
<Replace>
   <CmdID>3</CmdID>
   <Meta>
      <FieldLevel xmlns="syncml:metinf"/>
      <Type xmlns="syncml:metinf">x-type/x-subtype</Type>
   </Meta>
   <Item>
      <Target>
         <LocURI>244</LocURI>
      </Target>
      <Data>
...
      </Data>
```

```
    </Item>

</Replace>
```

5.2.4 Format

Usage: Specifies the encoding format of the content information in the Data element.

Parent Elements: `MetInf`

Restrictions: The value of this element SHOULD be one of bin, bool, b64, chr, int, node, null, xml, date, time, or float. If the element type is missing, the default value is chr. If the value is bin, then the format of the content is binary data. If the value is bool, then the format of the content is either true or false. If the value is b64, then the format of the content information is binary data that has been character encoded using the Base64 transfer encoding defined by [RFC2045]. If the value is chr, then the format of the content information is clear-text in the character set specified on either the transport protocol, the MIME content type header or the XML prolog. If the value is int, then the format of the content information is numeric text representing the integer. If the value is node, then the content represents an interior object in the management tree. If the value is null, then there is no content information. This value is used by some synchronization data models to delete the content, but not the presence of the property. If the value is xml, then the format of the content information is XML structured mark-up data. If the value is date, then the format of the content is in ISO 8601 format with the century being included in the year [ISO8601]. If the value is time, then the format of the content is in ISO 8601 format. If the value is float, then the format of the content is standard concept of real numbers corresponding to a single precision 32 bit floating point type as defined in XML Schema 1.0 as the `float` primitive type [XMLSCHEMADT].

In case a `Meta` element containing a `Format` element contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to.

The target object is the one in which the meta-information appears.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify format meta-information for data in the Item element type.

```
<Item>
  <Meta>
    <Format xmlns='syncml:metinf'>int</Format>
  </Meta>
  <Data>1024</Data>
</Item>
```

5.2.5 FreeID

Usage: Specifies the number of free item identifiers available for adding new items to a local datastore or device.

Parent Elements: `Mem`

Restrictions: Content information MUST be specified as the decimal integer number of item identifiers that are currently available for new items in the local datastore. If the Mem element type is present, then the FreeID element type is REQUIRED.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following is an example of 25 free item identifiers.

```
<FreeID xmlns='syncml:metinf'>25</FreeID>
```

5.2.6 FreeMem

Usage: Specifies the amount of free memory, in bytes, available in a local database or a device.

Parent Elements: `Mem`

Restrictions: Content information MUST be specified as the decimal integer number of free bytes of memory currently available in the local database. If the Mem element type is present, then the FreeMem element type is REQUIRED.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following is an example for 1022 free bytes.

```
<FreeMem xmlns='syncml:metinf'>1022</FreeMem>
```

5.2.7 Last

Usage: Specifies the synchronization state information (i.e., sync anchor) for the last successful synchronization session.

Parent Elements: `Anchor`

Restrictions: The value MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601].

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Anchor>
  <Last xmlns='syncml:metinf'>20000824T133000Z</Last>
  <Next xmlns='syncml:metinf'>20000824T221300Z</Next>
</Anchor>
```

5.2.8 Mark

Usage: Specifies a meta-information "mark" on the data object.

Parent Elements: `MetInf`

Restrictions: The content information for this element type SHOULD BE one of draft, final, delete, undelete, read, unread.

When this meta-information is specified repetitively in a hierarchically of element types (e.g., in a SyncML collection, as well as the items in the collection), then the meta-information specified in the lowest level element type takes precedence.

This element type is used to set the meta-information characteristics of a data object, such as the draft/final, delete/undelete, read/unread marks on a folder item or mail item.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about a data object specified by the Item element type.

```
<Update>
  <CmdID>10</CmdID>
  <Item>
    <Source>
      <LocName>jsmith</LocName>
      <LocURI>host1.com-19991208T234504-001</LocURI>
    </Source>
    <Meta>
      <Mark xmlns='syncml:metinf'>unread</Mark>
    <Meta>
  </Item>
</Update>
```

5.2.9 MaxMsgSize

Usage: Specifies the maximum byte size of any response message to a given SyncML request.

Parent Elements: `MetInf`.

Restrictions: The element type appears in the Meta element in the SyncHdr of a SyncML request to specify the maximum size of any subsequent response messages. The element type is usually specified by a SyncML client, but can also be specified by a SyncML server.

This element type value is applicable for the remainder of the synchronization session, unless it is specified again.

The element type value is the text string representation of the maximum, decimal byte size of any response message.

In order to use the elements from the MetInf name space, the root element does not need to be specified.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: Normally, the root element type does not appear in a SyncML Meta element type.

```
<MaxMsgSize xmlns='syncml:metinf'>1023</MaxMsgSize>
```

5.2.10 MaxObjSize

Usage: Specifies the maximum size in bytes of a data object that the device is able to receive.

Parent Elements: `MetInf`.

Restrictions: The element type appears in the Meta element of a SyncML request to specify the maximum size of the largest object it is capable of receiving in any subsequent response messages. This element type value is applicable for the remainder of the synchronization session.

The element type value is the text string representation of the maximum, decimal byte size without leading zeroes of any object.

In order to use the elements from the MetInf name space, the root element does not need to be specified.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: Device that can receive a maximum object of 10K bytes.

```
<MaxObjSize xmlns='syncml:metinf'>10240</MaxObjSize>
```

5.2.11 Mem

Usage: Specifies the maximum free memory and item identifier for a source (e.g., a datastore or a device.)

Parent Elements: `MetInf`

Restrictions: The element type is OPTIONAL.

Content Model:

```
(SharedMem?, FreeMem, FreeID)
```

Attributes: None.

Example: The following example specifies a shared datastore memory within a Sync command.

```
<Sync>
  <CmdID>1</CmdID>
  <Target><LocURI>./contacts/james_bond</LocURI></Target>
  <Source><LocURI>./dev-contacts</LocURI></Source>
  <Meta>
    <Mem xmlns='syncml:metinf'>
      <SharedMem xmlns='syncml:metinf'/>
      <FreeMem xmlns='syncml:metinf'>8100</FreeMem>
      <!--Free memory (bytes) in Contact database on a device -->
      <FreeID xmlns='syncml:metinf'>81</FreeID>
      <!--Number of free records in Contact database-->
    </Mem>
```

```
  </Meta>
...
</Sync>
```

5.2.12 MetInf

Usage: Specifies the root element for the SyncML meta-information document.

Parent Elements: `Root element type.`

Restrictions: In order to use the elements from the MetInf name space, the root element does not need to be specified. The element type can appear in the Meta element of a SyncML document to allow for declaring a default name space.

Content Model:

```
MetInf (FieldLevel?, Format?, Type?, Size?, Anchor?, Version?, NextNonce?,
MaxMsgSize?, MaxObjSize?, EMI*, Mem?)
```

Attributes: None.

Example: Normally, the root element type does not appear in a SyncML Meta element type.

```
<MetInf xmlns='syncml:metinf'>
  <FieldLevel/>
  <Type xmlns='syncml:metinf'>x-type/x-subtype</Type>
  <Format xmlns='syncml:metinf'>chr</Format>
  <Size xmlns='syncml:metinf'>877566</Size>
  <Version xmlns='syncml:metinf'>20000714T082300Z</Version>
</MetInf>
```

5.2.13 Next

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current synchronization session.

Parent Elements: `Anchor`

Restrictions: The value MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601].

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Anchor xmlns='syncml:metinf'>
  <Last xmlns='syncml:metinf'>20000824T133000Z</Last>
  <Next xmlns='syncml:metinf'>20000824T221300Z</Next>
</Anchor>
```

5.2.14 NextNonce

Usage: Specifies the nonce string to be used in any subsequent communication.

Parent Elements: `MetInf`

Restrictions: The nonce string MUST be further re-formatted using the Base64 algorithm. Terminators or length of Nonce String MUST NOT be included in this re-formatting. The Nonce string MUST be treated as opaque data.

This element type is used to specify the next nonce string that is to be used in any subsequent SyncML message. For example, a SyncML server specifies this element type to tell the SyncML client to change its nonce to a new value.

Nonce strings are used in the SyncML "MD5 Digest" scheme of authentication credentials.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Meta>
  <NextNonce
xmlns='syncml:metinf'>QWxhZGRpbjpvcGVuIHNlc2FtZQ==</NextNonce> </Meta>
```

5.2.15 SharedMem

Usage: Specifies if the datastore memory is shared. If the memory is shared, the actual memory space is used also by other datastores, and the actual memory space can be more limited than in theory it might be.

Parent Elements: `Mem`

Restrictions: The content of this element MUST be empty. This element type is used as a flag, and if this element type is present, then the given datastore memory is shared. This element is OPTIONAL.

Content Model:

```
EMPTY
```

Attributes: None.

Example: The following is an example of shared datastore memory.

```
<Mem xmlns='syncml:metinf'>
  <SharedMem xmlns='syncml:metinf'/>
  <FreeMem xmlns='syncml:metinf'>25601</FreeMem>
  <FreeID xmlns='syncml:metinf'>200</FreeID>
</Mem>
```

5.2.16 Size

Usage: Specifies the byte size of a data object.

Parent Elements: `MetInf`

Restrictions: The byte size is specified as the numeric text equivalent of the byte count of the data object. In case a `Meta` element containing a `Size` element contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to

Content Model:

```
(#PCDATA)
```

Attributes: None

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about the byte size of the Item element type.

```
<Item>
  <Target><LocURI>4</LocURI>
  <Meta>
    <Size xmlns='syncml:metinf'>10</Size>
  </Meta>
  <Data>John Smith</Data>
</Item>
```

5.2.17 Type

Usage: Specifies the media type of the content information in the Data element.

Parent Elements: `MetInf`

Restrictions: If this element is missing, then the default content-type is text/plain. The content information for this element type SHOULD BE a registered MIME content-type. Alternatively, a URN can be used to specify the media type. In case a `Meta` element containing a `Type` element contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to.

Content Model:

```
(#PCDATA)
```

Attributes: None

Example: The following example illustrates how the element type is used within a SyncML message to specify meta-information about the media type of the content information in the Item element type.

```
<Item>
  <Target><LocURI>3</LocURI></Target>
  <Meta>
    <Type xmlns='syncml:metinf'>text/directory;profile=vCard</Type>
  </Meta>
  <Data>BEGIN:VCARD
VERSION:3.0 FN:Jim Smith N:Smith;Jim TEL;TYPE=WORK,VOICE,FAX:+1-919-555-
1234 EMAIL;TYPE=INTERNET,WORK:Jim_Smith@mail.host.com
END:VCARD
  </Data>
</Item>
```

5.2.18 Version

Usage: Specifies the revision identifier of a data object.

Parent Elements: `MetInf`

Restrictions: The value MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601]. Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object. In case a `Meta` element containing a `Version` element contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to.

If not present, then the object doesn't currently have a revision version identifier. When the element type is missing, this SHOULD NOT be interpreted as meaning the original version of the data object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about the synchronization version of the Item element type.

```
<Item>
  <Target><LocURI>4</LocURI>
  <Meta>
    <Version xmlns='syncml:metinf'>20000301T133000Z</Version>
  </Meta>
  <Data>John Smith</Data>
</Item>
```

6. DTD Definition

```
<!--

SyncML Meta Information (SYNCML-METINF) V1.2 Document Type Definition
modified 03 Mar 2005


Copyright Open Mobile Alliance Ltd., 2002-2005

          All rights reserved


This DTD defines a sequence of meta-information that is used within

the SyncML Representation Protocol DTD. Typical usage:

   <!DOCTYPE MetInf PUBLIC "-//OMA//DTD SYNCML-METINF 1.2//EN"

             "http://www.openmobilealliance.org/tech/DTD/OMA-TS-
SyncML_MetaInfo_DTD-V1_2.dtd"

             [<?oma-syncml-metinf-ver supported-versions="1.2"?>]>

   <element>

      ...

   </element>


Terms and conditions of use are available from the

Open Mobile Alliance Ltd. web site at

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

-->


<!-- Root Element -->

<!ELEMENT MetInf (FieldLevel?, Format?, Type?, Mark?, Size?, Anchor?,
Version?, NextNonce?, MaxMsgSize?, MaxObjSize?, EMI*, Mem?)>

<!-- FieldLevel change flag -->

<!ELEMENT FieldLevel EMPTY>

<!-- Format or encoding type -->

<!ELEMENT Format (#PCDATA)>

<!-- Element specific type specification -->
```

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

<!-- Mark -->

<!ELEMENT Mark (#PCDATA)>

<!-- Byte count -->

<!ELEMENT Size (#PCDATA)>

<!-- Data versioning info -->

<!ELEMENT Anchor (Last?, Next)>

<!ELEMENT Last (#PCDATA)>

<!ELEMENT Next (#PCDATA)>

<!ELEMENT Version (#PCDATA)>

<!ELEMENT NextNonce (#PCDATA)>

<!ELEMENT MaxMsgSize (#PCDATA)>

<!ELEMENT MaxObjSize (#PCDATA)>

<!-- Experimental Meta Information extension -->

<!ELEMENT EMI (#PCDATA)>

<!-- Dynamic Memory -->

<!ELEMENT Mem (SharedMem?, FreeMem, FreeID)>

<!-- Free Memory in the number of identifiers -->

<!ELEMENT FreeID (#PCDATA)>

<!-- Free Memory in bytes -->

<!ELEMENT FreeMem (#PCDATA)>

<!-- Shared Memory -->

<!ELEMENT SharedMem EMPTY>

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

7. WBXML Definition

The following tables define the token assignments for the mapping of the Meta-Information DTD element types into WBXML as defined by [WBXML].

7.1 Code Space and Code Page Definitions

The element types from the Meta-Information DTD (i.e., name space) are only intended to be used within a SyncML document. Hence, the elements from the Meta-Information DTD are always mapped into the SyncML WBXML code space.

DTD Name	WBXML PUBLICID Token (Hex Value)	Formal public Identifier
SyncML	0x1202	-//SYNCML//DTD SyncML 1.2//EN

The SyncML DTD is assigned the WBXML document public identifier (i.e., the "publicid" WBXML BNF production) associated with the 0x1202 token.

The element types from the Meta-Information DTD utilize the code page x01 (one) within the SyncML Code Space.

DTD Name	WBXML Code Page Token (Hex Value)	Formal Public Identifier
SyncML	00	-//SYNCML//DTD SyncML 1.2//EN
MetaInf	01	-//SYNCML//DTD MetInf 1.2//EN

7.2 Token Definitions

The following WBXML token codes represent element types (i.e., tags) form code page x01 (one), Meta-Information DTD.

Element Type Name	WBXML Tag Token (Hex Value)
Anchor	05
EMI	06
Format	07
FreeID	08
FreeMem	09
Last	0A
Mark	0B
MaxMsgSize	0C
Mem	0D
MetInf	0E

Next	0F
NextNonce	10
SharedMem	11
Size	12
Type	13
Version	14
MaxObjSize	15
FieldLevel	16

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-SyncML-MetaInfo-V1_1_2-20030612-A	12 Jun 2003	Initial OMA release
OMA-TS-SyncML-MetaInfo-V1_2-20070221-A	21 Feb 2007	Status changed to Approved by TP: OMA-TP-2007-0005R03-INP_ERP_SyncML_Common_V1_2_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in **Error! Reference source not found.**.

B.1 Client Meta Information

Item	Function	Ref.	Status	Requirement
DSDM-METINF-C-001	Support for Anchor element	5.2.1	M	DSDM-METINF-C-007 AND DSDM-METINF-C-013
DSDM-METINF-C-002	Support for EMI element	5.2.2	O	
DSDM-METINF-C-003	Support for FieldLevel element	5.2.3	O	
DSDM-METINF-C-004	Support for Format element	5.2.4	M	
DSDM-METINF-C-005	Support for FreeID element	5.2.5	O	
DSDM-METINF-C-006	Support for FreeMem element	5.2.6	O	
DSDM-METINF-C-007	Support for Last element	5.2.7	M	
DSDM-METINF-C-008	Support for Mark element	5.2.8	O	
DSDM-METINF-C-009	Support for MaxMsgSize element	5.2.9	O	
DSDM-METINF-C-010	Support for MaxObjSize element	5.2.10	O	
DSDM-METINF-C-011	Support for Mem element	5.2.11	O	DSDM-METINF-C-005 AND DSDM-METINF-C-006 AND DSDM-METINF-C-015
DSDM-METINF-C-012	Support for MetInf element	5.2.12	M	DSDM-METINF-C-001 AND DSDM-METINF-C-004 AND DSDM-METINF-C-014 AND DSDM-METINF-C-017 AND DSDM-METINF-C-018
DSDM-METINF-C-013	Support for Next element	5.2.13	M	
DSDM-METINF-C-014	Support for NextNonce element	5.2.14	M	
DSDM-METINF-C-015	Support for SharedMem element	5.2.15	O	
DSDM-METINF-C-016	Support for Size element	5.2.16	O	
DSDM-METINF-C-017	Support for Type element	5.2.17	M	
DSDM-METINF-C-018	Support for Version element	5.2.18	M	

B.2 Server Meta Information

Item	Function	Ref.	Status	Requirement
DSDM-METINF-S-001	Support for Anchor element	5.2.1	M	DSDM-METINF-S-007 AND DSDM-METINF-S-013
DSDM-METINF-S-002	Support for EMI element	5.2.2	O	
DSDM-METINF-S-003	Support for FieldLevel element	5.2.3	O	
DSDM-METINF-S-004	Support for Format element	5.2.4	M	
DSDM-METINF-S-005	Support for FreeID element	5.2.5	M	
DSDM-METINF-S-006	Support for FreeMem element	5.2.6	M	
DSDM-METINF-S-007	Support for Last element	5.2.7	M	
DSDM-METINF-S-008	Support for Mark element	5.2.8	O	
DSDM-METINF-S-009	Support for MaxMsgSize element	5.2.9	M	
DSDM-METINF-S-010	Support for MaxObjSize element	5.2.10	M	
DSDM-METINF-S-011	Support for Mem element	5.2.11	M	DSDM-METINF-S-005 AND DSDM-METINF-S-006 AND DSDM-METINF-S-015
DSDM-METINF-S-012	Support for MetInf element	5.2.12	M	DSDM-METINF-S-001 AND DSDM-METINF-S-004 AND DSDM-METINF-S-009 AND DSDM-METINF-S-010 AND DSDM-METINF-S-011 AND DSDM-METINF-S-014 AND DSDM-METINF-S-017 AND DSDM-METINF-S-018
DSDM-METINF-S-013	Support for Next element	5.2.13	M	
DSDM-METINF-S-014	Support for NextNonce element	5.2.14	M	
DSDM-METINF-S-015	Support for SharedMem element	5.2.15	M	
DSDM-METINF-S-016	Support for Size element	5.2.16	O	
DSDM-METINF-S-017	Support for Type element	5.2.17	M	
DSDM-METINF-S-018	Support for Version element	5.2.18	M	