SyncML Representation Protocol, version 1.0

Abstract

SyncML is a specification for a common data synchronization framework and XML-based format, or representation protocol, for synchronizing data on networked devices. SyncML is designed for use between mobile devices that are intermittently connected to the network and network services that are continuously available on the network. SyncML can also be used for peer-to-peer data synchronization. SyncML is specifically designed to handle the case where the network services and the mobile device store the data they are synchronizing in different formats or use different software systems.

SyncML Initiative

The following companies are Sponsors of the SyncML initiative:

Ericsson
IBM
Lotus
Matsushita Communications Industrial Co., Ltd.
Motorola
Nokia
Palm, Inc.
Psion
Starfish Software

Revision History

Revision	Date	Comments
v0.9	2000-05-31	Changed DTD. Sync initiated with Sync, not Alert. Final flag at end of SyncBody, not in SyncHdr. Added TargetRef and SourceRef to Results. Completed command descriptions. Added WBXML token definitions. Added Data to Search command.
V0.91	2000-07-07	Added request status code for sync retry needed. Removed DevInf and MetInf WBXML as this will be defined in the individual DTD specifications. Removed section 15 as was superfluous. Added placeholder in WBXML definitions for the xMsg code page. Corrected namespace usage typos. Added stactic conformance requirements section. Revised SCR section. Added URI based search/filtering for contacts, calendar/task and email objects.
V0.92	2000-08-11	Added success code 211 for the "SftDel-HrdDel" condition. Clarifies some Status code text for "update conflict" states. Changes to the SRC tables in section 9. Removed undefined common media types.

V1.0 Alpha	2000-09-04	Updated WBXML encoding Authentication challenge functionality added. Modifications at various places because of addition. Slight revision of text in security section. Removed "magic cookie" in MIME registration sections. CmdID required on all commands but Status. Updated Alert value table descriptions and aligned values with current Sync Proto spec. Update command SRC for client-sending changed to SHOULD. SourceRef/TargetRef MUST be specified in a Status when Source/Target specified in corresponding SyncML command. Fixed Sponsor names one mo' time. Added 512 status code for unsupported sync protocol version. Changes in CmdID, CmdRef, Status to make CmdID=0 refer to the SyncHdr. CmdRef and MsgRef now required in Status. Added explanatory text to Status to explain the "Status on a Status" usage case. Took out remaining "editors notes". Separated Update and Replace Syncml commands. Temporarily removed Update command until we get command specification fixed. 09/04/00: REMOVED REPLACE ELEMENT FROM WITHIN REPLACE CONTENT MODEL IN DTD
V1.0 Beta	2000-11-13	Added explaination for TargetRef and SourceRef in Section 4.17. Removed reference to Create/Delete in Section 5.5.8. Changed text in Section 5.5.5 to explain how a delete operation is accomplished for two-step deletion applications such as that found in email applications with a special "Deleted" folder. Added Meta to SyncHdr to allow inclusion of the MaxMsgSize meta-information. Modified text in Replace chapter for case when Replace acts as an Add command. Changed/improved text for 419 status code. Clarified in Section 5.4.1 that the ordering of Status commands MUST match the order of the associated commands in the SyncML Request. Clarified that NoResp on Atomic, Sequence or Sync specify no Status for just the corresponding Atomic, Sequence or Sync. Changed MIME type registration form to show "+xml" and "+wbxml", to be consistent with pending IETF RFC on MIME naming for XML entities. Clarified use of relative URI in LocURI element type. Removed duplicate section on URI CGI script filterning. Also added text to URI CGI script filetering for Get/Put on an arbitrary database or XML document. Supporter comments received to date.
V1.0c	2000-11-24	SftDel Chapter updated, Session ID, Put, and Get items in the SCR table clarified, alert codes updated to match with the sync protocol spec, SyncML DevInf, xCard, xCal, and xBookmark removed from the WBXML code space of SyncML DTD, the code page of Meta Info DTD changed, the DevInf MIME type introduced in the base media types
V1.0	2000-12-07	The candidate version for the final release.

Copyright Notice

Table of Contents

1 Introduction

SyncML is a specification for a common data synchronization framework and XML-based format, or representation protocol, for synchronizing data on networked devices. SyncML is designed for use between mobile devices that are intermittently connected to the network and network services that are continuously available on the network. SyncML can also be used for peer-to-peer data synchronization. SyncML is specifically designed to handle the case where the network services and the device store the data they are synchronizing in different formats or use different software systems.

The SyncML representation protocol is defined by a set of well-defined messages that are conveyed between entities participating in a data synchronization operation. The messages are represented as an XML document. XML is the industry standard for text document mark-up, as defined in [14].

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 data synchronization 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 data synchronization operations. This conceptual data synchronization "package" permits either a "batch" of multiple data synchronization operations put together in a single SyncML Message or conveyed as separate SyncML Messages, each containing a single data synchronization operation. SyncML Messages are the body of the MIME entities.

2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interoperated as described in [5].

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

3 Terminology

This section defines terminology used throughout the specification.

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 data synchronization role when the application issues SyncML "request" messages. For example, the **`Sync`** SyncML Command in a SyncML Message.

command

A SyncML Command is a data synchronization 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 synchronization data and meta-information. The SyncML Message is an XML document.

operation

A SyncML Operation refers to the conceptual data synchronization transaction achieved by the SyncML Commands specified by a SyncML Package. For example, "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 data synchronization 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 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 data synchronization role when an application issues SyncML "response" messages. For example, a Results Command in a SyncML Message.

4 SyncML

The SyncML representation protocol does not specify the data synchronization protocol or "sync engine", but rather specifies a common synchronization framework and format that accommodates different data synchronization models. The SyncML representation protocol specifies what the result of the various synchronization operations must be. A common data synchronization protocol can be defined using the SyncML representation protocol. This is the subject of another specification.

4.1 SyncML Framework

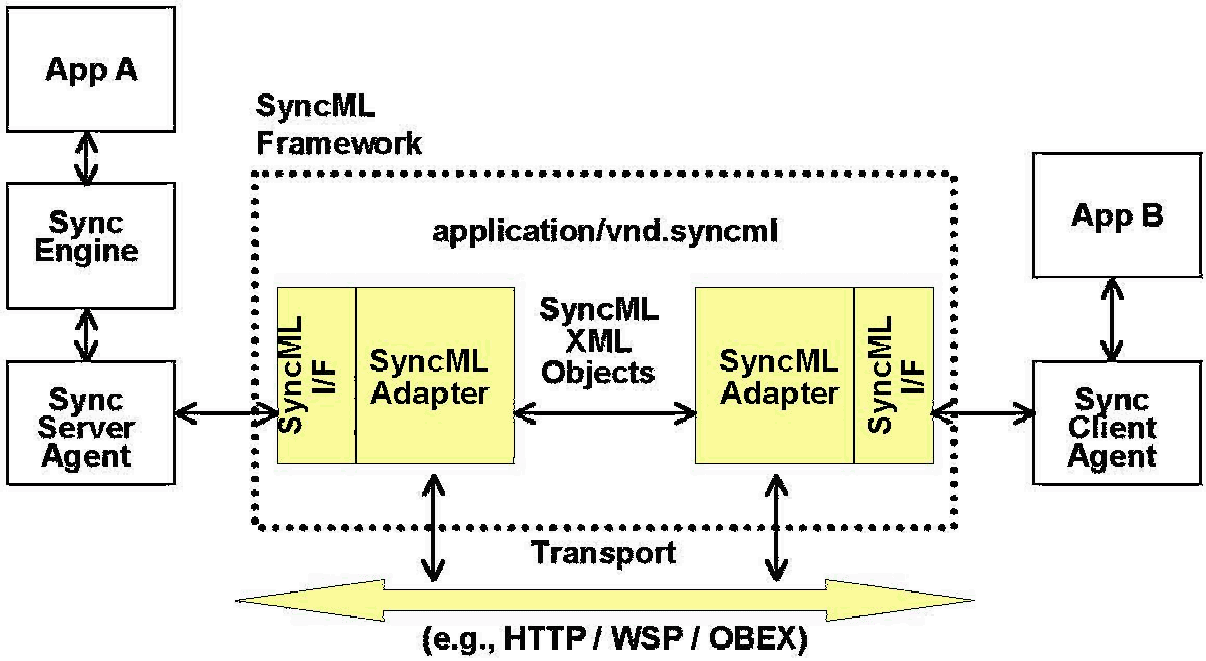
SycnML not only defines a format, but also a conceptual data synchronization framework and data synchronization protocol. The framework is depicted in Figure 1. In the figure, the scope of the SyncML Framework is show by the dotted-line box. The Framework consists of the SyncML representation protocol, as well as a conceptual SyncML Adapter and SyncML Interface. This SyncML Framework is useful for describing the particular system model associated with SyncML implementations.

The SyncML data synchronization protocol is outside the SyncML Framework, but is essential for providing interoperable data synchronization. The SyncML data synchronization protocol is defined by another, companion SyncML specification [11].

The application "A" depicts a networked service that provides data synchronization with other applications, in this case application "B", on some networked device. The service and device are connected over some common network transport, such as HTTP. Application "A" utilizes a data synchronization protocol, implemented as the "Sync Engine" process. The data synchronization protocol is manifested on the network by client applications accessing the "Sync Server" network resource. The "Sync Server Agent" manages the "Sync Engine" access to the network and communicates the data synchronization operations to/from the client application. The "Sync Server Agent" performs these capabilities through invocations to functions in the "SyncML I/F" or interface. The "SyncML I/F" is the application programming interface to the "SyncML Adapter". The "SyncML Adapter" is the conceptual process that the originator and recipient of SyncML formatted objects utilize to communicate with each other. The "SyncML Adapter" is also the framework entity that interfaces with the network transport, which is responsible for creating and maintaining a network connection between Application "A" and Application "B". Application "B" utilizes a "Sync Client Agent" to access the network and it's "SyncML Adapter", through invocations of functions in the "SyncML I/F".

Actual server and client implementations may not be implemented in the discrete components identified by this conceptual framework. However, this framework is useful for a common discussion of the components that are necessary to implement a common data synchronization protocol.

Figure 1: SyncML Framework



4.2 SyncML Package and Messages

In SyncML, the data synchronization operations are conceptually bounded 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 data synchronization 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 synchronization data or meta-information.

4.3 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 a synchronization data accessible to the recipient. This command MUST only be specified within a `Sync` command.

- `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 SHOULD 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. The deletion can either be a soft- or hard-delete.

- `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. This command MUST NOT be specified within a `Sync` command.

- `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. This command MUST NOT be specified within a `Sync` command.

- `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. This command MUST only be specified within a `Sync` command

- `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 SHOULD 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 synchronized. 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.

In SyncML, a data collection, **A**, has been successfully synchronized with a data collection, **B**, if and only if:

There exists a bijective mapping **M**, such that for all identifiers **I**:

if **A(I)** exists, then **B(M(I))** exists, and **B(M(I))** is equivalent to **A(I)**.

if **A(I)** does not exist, then **B(M(I))** does not exist.

If (**A** synchronized with **B**) is true, this does not imply that (**B** synchronized with **A**) is also true. In other words, data synchronization using SyncML does not have to be reflexive.

4.4 SyncML Data Formats

SyncML not only provides for a common set of commands, but also identifies a small set of common data formats. The data formats provide a common set of media types for exchanging common accepted information, such as contacts, calendars and messages. Support for these data formats is mandatory for conformance to this specification. In addition to these common formats, SyncML allows for the identification of any other registered format. SyncML utilizes the MIME content type framework for identifying data formats, called MIME media types.

4.5 Capabilities Exchange

SyncML supports capabilities exchange. Capabilities exchange is the ability of a SyncML Client and Server to determine what device, user and application features each supports. The capabilities exchange, from the SyncML Server perspective, is achieved by using the `Get` command to retrieve the device information, user information and application information documents from the SyncML Client. The capabilities exchange, from the SyncML Client perspective, is achieved by using the `Get` command to retrieve the analogous documents from the SyncML Server. These documents contain profile information about support for well-defined features. In addition, the `Put` command can be used to by the SyncML Client to push capabilities exchange information to the SyncML Server.

The capabilities exchange can also be used to establish or administer SyncML data synchronization services between a SyncML Client and Server.

Refer to [9] for further details on the specification of the Device Information DTD.

4.6 Data Identifier Mapping

SyncML does not require that two data stores being synchronized be of the same schema (i.e., aren't homogeneous). Specifically, SyncML allows for both the data identifiers and the data formats to be different in the two data collections. However, in such cases in order to use SyncML, the synchronizing applications would need to provide a mapping between data identifiers in one data store and those in another. For example, a document on the data synchronization server could be identified with a 16 byte, globally unique identifier (GUID). The corresponding version of this document on a mobile device could be identified by a small, two byte, and local unique identifier (LUID). Hence, to synchronize the data on the mobile device with the data on the data synchronization server, the synchronizing application would have to map the smaller identifiers of the mobile device to the larger identifiers used by data synchronization server; and visa versa. SyncML includes the necessary mechanism to specify such an identifier mapping.

4.7 Refreshing Data

In addition to synchronization, SyncML includes commands that are not normally thought of as synchronization operations, but are still required in a practical data synchronization protocol. For example, SyncML provides the capability for refreshing the entire data on the SyncML client with the equivalent synchronization data on the SyncML server. This may be necessary if the SyncML client and the SyncML server versions are no longer "in sync" with each other due to a hardware or power failure in the mobile device, or if the version on the SyncML client has become corrupted or erased from memory. This capability is provided by the SyncML client issuing a "refresh" Alert command to the SyncML server.

4.8 Soft and Hard Data Deletion

The SyncML `Delete` command provides the capability for a SyncML request to delete data from the recipient's data store. Two forms of deletion are supported. Normally, when a `Delete` command is specified, it conveys a request to completely delete the specified data from the recipient's data store. The deleted data SHOULD no longer be associated with the originator's synchronization data. This is the semantics of a "Hard Delete". In addition, SyncML provides support for a "Soft Delete" command. The "Soft Delete" provides the capability to request the deletion of data from the recipient but NOT to remove it from the recipient's synchronization data. The rationale for a "Soft Delete" is based on the possibility of limited storage resources in a mobile device. The data is deleted to free-up storage for other, higher priority data. Conceptually, the deleted data remains in the set of synchronized data for the recipient of the "Soft Delete" command.

On occasions, an exception can occur where a data element on the SyncML client is "Soft Deleted" and the same data element is "Hard Deleted" on the SyncML server. This condition will cause a "Soft-Delete Conflict" for that event when a two-way synchronization is attempted. This version of SyncML does not specify how to negotiate the resolution of such "Soft-Delete Conflicts". However, it does provide status codes to identify Soft-Delete Conflict conditions and to also identify how the conflict might have been resolved.

4.9 Archiving Data

The SyncML `Delete` command provides the capability for SyncML request to data on the recipient prior to deleting it from the device. This is indicated by the presence of the `Archive` element type in the `Delete` command. Some recipient's might not support this feature. In which case, the would generate an error condition (i.e., `(406) Optional feature not supported.`).

4.10 Replacing Data

The SyncML `Replace` command provides the capability for the originator to replace existing data. The command can also be the cause for an "Update Conflict".

In addition, the SyncML `Replace` command provides the capability for the originator to update and replace meta-information in the target data element. For example, an item in an email inbox that is marked as `Unread` can be changed to be marked as `Read`. This capability is specified by the presence of the `Mark` element type in the `Meta` element type within a `Replace` command. Refer to [10] for further details on the Meta Information DTD specification.

On occasions, an exception can occur where the same data element on both the SyncML client and the SyncML server have been updated or replaced. For example, the start and end date/time for the same event might have been changed to different values on the SyncML client compared to the description on the SyncML server. This condition will cause an "Update Conflict" for that event when a two-way synchronization is attempted. This version of SyncML does not specify how to negotiate the resolution of such Update Conflicts. However, it does provide status codes to identify Update Conflict conditions and to also identify how the conflict might have been resolved.

4.11 Searching For Data

The SyncML `Search` command provides the capability for searching a recipient data store for particular data. This command provides support for any registered search grammar. The specific search grammar is identified by the `Type` element in the `Meta` element type within the Search command.

The SyncML `Search` command can be used to select items within a data store to be used as the source for a subsequent SyncML `Sync` Command.

In addition, SyncML enables a search or filter to be specified on the `Target LocURI` element within the `Sync` command. With this capability, SyncML clients can specify filter constraints on the database records for a `Sync` command. For example, a mobile client can specify that the synchronization with a server calendar database is restricted to today's events.

4.12 Localization

The SyncML representation protocol allows an originator to specify the desired localization for the synchronization operation and synchronization data for any registered language. The `xml:lang` attribute can be specified on any element type to identify the language used

in the element type's content model. In addition, a `Get` and `Search` command can specify the desired language for results by specifying the `Lang` element type in the `Get` or `Search` command.

The default character set for SyncML representation protocol is UTF-8, as defined in [7].

4.13 Security

An objective of SyncML is to provide a framework for secure data synchronization. SyncML itself does not define any new security schemes. Instead, it provides the framework for 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 [4], 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`. This authentication scheme is a MD5 digest form of the concatenation of the an authentication identifier such 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, followed by a recipient specified nonce string. The maximum duration that the nonce string can be used by the originator is the current SyncML 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 [3]. The MD5 credential value MUST also 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 Sync protocol are defined in [11].

4.14 XML Usage

The SyncML Messages are represented in a mark-up language defined by [14]. 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 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, `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_v10_20001207.dtd` or the URN `syncml:syncml`. The SyncML DTD are also identified by the ISO 9070 formal public identifier `-//SYNCML//DTD SyncML 1.0//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 [13]. The use of [13] 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 it's content. This is extremely useful for applications such as data synchronization, where not just content, but structure semantics is often exchanged.

4.15 MIME Usage

The [4] 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. There are two MIME content types for the SyncML Message. The MIME content type of `application/vnd.syncml+xml` identifies the clear-text XML representation for the SyncML Message. The MIME content type of `application/vnd.syncml+wbxml` identifies the WBXML binary representation for the SyncML Message. Section 10 of this specification specifies the MIME content type registration for these two MIME media types.

One of these two MIME content types MUST BE used for identifying SyncML Messages within transport and session level protocols that support MIME content types.

4.16 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 [8], 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 [14], an application will need to confirm the validity of any URI or URN.

SyncML uses the `IMEI` URN type to identify an International Mobile Equipment Identifiers [16]. The IMEI URN specifies a valid, 15 digit IMEI. In addition, SyncML uses the `SYNCML` URN type to identify SyncML specific name spaces and unique names. Other URN types may be used in the `LocURI` element type also.

4.17 Target and Source Addressing

The `Target` and `Source` element types are used in SyncML to specify target and source routing addresses, respectively, within the `LocURI` element type. The `LocURI` SHOULD be either a location URI or location URN, but in certain cases can also be a locally unique identifier (see the table below). In addition, an optional display name (i.e., `LocName`) can be specified within the `Target` and `Source` element types to provide a display name for the `LocURI` value.

The semantics of the `LocURI` value is context specific. That is, the location routing address has usage that is specific to the command in which it appears. For instance, in an `Alert`, the `LocURI` value in the `Target` element type addresses a database that is the target of the alert message. Or for another instance, in a `MapItem`, the `LocURI` value addresses the identifier for an individual item in a local database.

Where a URI is specified, either an absolute or relative URI value MUST be used. The only time a relative URI MUST be used is when the information in the SyncML message is sufficient for a recipient to construct the absolute URI. For example, the relative URI in the `Target` element in an `Alert` command can be converted to the proper absolute URI by prefixing the value from the `Target` element type found in the `SyncHdr`.

The following table specifies what the expected value and usage is in each of the contexts for `Target` and `Source` element types within a SyncML document.

Element	Contextual Address Requirement
SyncHdr	
`Target and Source`	Specifies the address of either the data synchronization server or the mobile client. When addressing the: *Server* - MUST use LocURI element type to specify the absolute URI form of network address of the synchronization server. For an absolute LocURI value, CGI script parameters can be appended to the URI to perform selection filtering on the server target. *Client* - MUST use LocURI element type to specify the absolute URI form of network address of the mobile client or IMEI URN to specify the global identifier associated with the mobile client.
`RespURI`	Specifies the address of to be used in the Target of the response message. The value is an absolute URI. CGI script parameter can be appended to the URI to perform selection filtering such as specifying a date/time after which the response should be performed.
Sync	
`Target and Source`	Specifies the address of either the server database or the local mobile client database. A relative URI can be specified, if the proper absolute URI can be constructed from prefixing the respective Target or Source value from the SyncHdr to this relative URI. When addressing the: *Server* Database - MUST use LocURI element type to specify either the absolute or relative URI for the server database. *Client* Database - MUST use LocURI element type to specify either the absolute or relative URI for the client database. For a LocURI value, CGI script parameters can be appended to the URI to perform selection filtering on the server target.
Search	
`Target`	If present, specifies the address on the recipient where the search results are to be temporarily stored. A relative URI can be specified, if the proper absolute URI can be constructed from prefixing the respective Target or Source value from the SyncHdr to this relative URI. When addressing the:

	Server - MUST use LocURI element type to specify the local URI where the search results are to be stored. *Client* - MUST use LocURI element type to specify either the absolute or relative URI where the search results are to be stored. For a LocURI value, CGI script parameters can be appended to the URI to perform selection filtering on the server target.
`Source`	Specifies one or more addresses on the recipient that are to be searched. When addressing the: *Server* - MUST use LocURI element type to specify the absolute or relative URI of the databases to be searched. *Client* - MUST use LocURI element type to specify the absolute or relative URI of the databases to be searched.
`Map`	
`Target`	Specifies the recipient database for the Map definition. A relative URI can be specified, if the proper absolute URI can be constructed from prefixing the respective Target or Source value from the SyncHdr to this relative URI. When addressing the: *Server* - MUST use LocURI element type to specify the absolute or relative URI of the server database. *Client* - MUST use LocURI element type to specify the mobile client database.
`Source`	Specifies the originator database for the Map definition. When addressing the: *Server* - MUST use LocURI element type to specify the absolute or relative URI of the server database. *Client* - MUST use LocURI element type to specify the mobile client database.
`MapItem`	
`Target`	Specifies the recipient item identifier. When addressing the: *Server* - MUST use LocURI element type to specify the locally unique identifier of the server database item. *Client* - MUST use LocURI element type to specify the locally unique identifier of the mobile client item.
`Source`	Specifies the originator item identifier. When addressing the: *Server* - MUST use LocURI element type to specify the

	locally unique identifier of the server database item. *Client* - MUST use LocURI element type to specify the locally unique identifier of the mobile client database item.
Item in an Alert, Exec, Get, Put Commands	
Target and Source	Specifies the address of the database item that is the argument of the SyncML command. When addressing the: *Server* - MUST use LocURI element type to specify the relative URI or URN for the server database. *Client* - MUST use either LocURI element type to specify the relative URI of the mobile client database. To address individual items within a database using the Get or Put command, URI addressing filtering techniques MUST be used.
Item in Add, Copy, Delete, Replace, Results, Status Commands	
Target and Source	Specifies the address of the database item that is the argument of the SyncML command. When addressing the: *Server* - MUST use LocURI element type to specify the absolute URI, relative URI, URN or locally unique identifier for the server database item. *Client* - MUST use either LocURI element type to specify the absolute URI, relative URI, URN or locally unique identifier for the client database item.

The Target and Source addresses are referenced in the `TargetRef` and `SourceRef` element types, respectively. When present, the `TargetRef` element type contains the value that was in the `Target` element type in a corresponding command. Respectively, the `SourceRef` element type contains the value that was in the `Source` element type in a corresponding command. For example, the `TargetRef` and `SourceRef` in a `Status` contain the respective `Target` and `Source` values from the command corresponding to the `Status`.

4.18 Target Address Filtering

Target address filtering is provided as a capability of this specification because it is considered to be easier to implement in mobile devices than the more robust Search command. Target address filtering is specified with CGI scripting.

The address filtering is restricted to the value of the `Target LocURI` element type. This type of target filtering is specified instead of using the Search command. Both forms of filtering SHOULD NOT be used in the same SyncML message.

Such CGI scripting can be used in the `Target LocURI` element type in the Sync command to filter or restrict the database records for the synchronization. The filter is

specified as a search part of the URI value that can be specified in the `Target LocURI` element type. The search part is the string found after the QUESTION MARK character (UTF-8 hexadecimal value 3F) in the URI.

Non-standard CGI scripting SHOULD NOT be used. When an invalid, unknown CGI script is found in a `LocURI` element type the error (`422`) `Bad CGI Script` MUST be returned in the Status command. In addition, the invalid portion of the CGI script SHOULD be returned in the `Item` element type of the `Status` command.

When CGI scripting in the `LocURI` element type is not supported, then when a CGI script is found in a `LocURI` element type, the error (`416`) `Optional feature not supported` MUST be returned in the `Status` command.

In the following example the calendar database events are filtered to just those that have an effectively start on or after 2000-07-14 and end no later than midnight 2000-07-14.

```
<Target>
    <LocURI>./mail/bruce1?DTSTART&GE;20000714T000000&AND;DTEND&LT;20000715
T000000</LocURI>
</Target>
```

The format for the CGI scripting is defined here in a formal notation, or ABNF, as defined in [15].

The CGI scripting is media type specific. However, there are some common CGI scripting used across the media types as specified in the following ABNF. If CGI scripting is supported, then all the logical CGI scripting primitives in the following table MUST be supported. Only two levels of logical MUST be supported (i.e., contains one logical separator). For example, "`FN&EQ;Frank%20Dawson&AND;TEL&CON;919`".

```
Log-op      = "&EQ;"    ;Logical Equal To
            / "&GT;"    ;Logical Greater Than
            / "&GE;"    ;Logical Greater Than Or Equal To
            / "&LT;"    ;Logical Less Than
            / "&LE;"    ;Logical Less Than Or Equal To
            / "&NE;"    ;Logical Not Equal To
            / "&CON;    ;Contains the value

log-sep     = "&OR;"    ;Logical OR
            / "&AND;"   ;Logical AND

exp-tag     = 1*VCHAR   ;Any experimental or other wise registered extension
string-value = 1*VCHAR ;Case sensitive string value

VCHAR       = %x21-7E / SPACE     ;Visible latin characters within UTF-8
                                  ;plus the SPACE character

SPACE       = %20
```

4.18.1 Arbitrary Database Object Filter

The filter for accessing individual items from an arbitary database object makes use of the script tags as specified by the following ABNF.

```
db-script = db-filter-item *(log-sep db-filter-item)
```

```
db-filter-item = "LUID" log-op luid-value

luid-value = string-value
 ;MUST be a valid locally unique identifier for
 ;an item in the database addressed by the LocURI
```

The following is an example of a value for a `Target LocURI` element type in a `Get` command that retrieves just the data item in a database corresponding to the locally unique identifier "16".

```
<Get>
  <CmdID>1</CmdID>
  <Item>
    <Target>
      <LocURI>./calendar?LUID=16</LocURI>
    </Target>
  </Item>
</Get>
```

4.18.2 XML Document Filter

The filter for accessing individual value for an element type or attribute list from an XML document object makes use of the script tags as specified by the following ABNF.

```
xml-script = xml-filter-item *(log-sep xml-filter-item)

xml-filter-item = "PROP" "&EQ;" xml-prop-name

xml-prop-name = string-value
 ;MUST be a valid element type name or attribute list name in the XML
document addressed by the LocURI
```

The following is an example of a value for a `Target LocURI` element type in a `Get` command that retrieves just the DBMem element type value in the XML document specified by the LocURI.

```
<Get>
  <CmdID>2</CmdID>
  <Item>
    <Target>
      <LocURI>./devinf10?PROP="DBMem"</LocURI>
    </Target>
  </Item>
</Get>
```

4.18.3 Contacts Media Object Filter

The filter for contact media objects makes use of the vCard CGI script tags conformant to the following ABNF.

```
Contact-script = contact-filter-item *(log-sep contact-filter-item)

contact-filter-item = vcard-tag log-op vcard-value

contact-tag = "FN" / "FAMILY" / "GIVEN" / "TEL" / "EMAIL" / exp-tag

contact-value = string-value ;MUST be a valid value for the vCard property
```

The values of "FN", "FAMILY", "GIVEN" are any string of case sensitive, visible UTF-8 characters. The values of "TEL" are any valid phone number. The values of "EMAIL" are any valid RFC822 email address string. If CGI scripting is supported for this object type, then all of these CGI scripting primitives MUST be supported.

The following is an example of a value for a Target LocURI element type that filters the synchronization of the contacts target database to all those records that contain the family name "Smith". The search values for "FAMILY" are case sensitive.

```
<Target>
  <LocURI>./mail/bruce9?FAMILY&EQ;Smith</LocURI>
</Target>
```

4.18.4 Calendar Media Object Filter

The filter for calendar media objects makes use of the iCalendar CGI script tags conformant to the following ABNF.

```
Cal-script = cal-filter-item *(log-sep cal-filter-item)

cal-filter-item = cal-tag log-op cal-value

cal-tag    = "START" / "END" / "DUE" / "LOCATION" / "SUMMARY"
           / "DESCRIPTION" / "ORGANIZER" / "ATTENDEE" / "METHOD"
           / "CATEGORIES" / "CLASS" / "PRIORITY" / "STATUS" / "TRANSP"
           / "RECURID" / "UID" / "URL" / exp-tag
cal-value = string-value ;MUST be a valid value for the iCalendar property
```

The values of "START", "END", "DUE" are formatted consistent with the complete representation, basic format for a calendar date and time of day as specified in [2]. The values of "RRULE" are a recurrence rule as specified in [1]. The values of "LOCATION", "SUMMARY", "DESCRIPTION", "CATEGORIES" are any string of case sensitive, visible UTF-8 characters. The values of "METHOD", "CLASS", "PRIORITY", "STATUS", "TRANSP" any of the valid enumerated values as specified in [1]. The value of "UID", "URL", "RECURID" are any valid value as specified for these properties in [1]. The values of "ORGANIZER" and "ATTENDEE" are any valid MAILTO URL. If CGI scripting is supported for the event form of this object type, then the "START" and "END"CGI scripting primitives MUST be supported. If CGI scripting is supported for the to-do form of this object type, then the "DUE" and "PRIORITY" CGI scripting primitives MUST be supported. If CGI scripting is supported for the journal entry form of this object type, then the "START", "END" and "DESCRIPTION" CGI scripting primitives MUST be supported.

The following is an example of a value for a Target LocURI element type that filters the synchronization of the target calendar database to all those records that contain a SUMMARY or DESCRIPTION with the string "Project XXX Review". The SPACE character must be specified by the hexadecimal encoding, as required by the format specification for Uniform Resource Identifiers.

```
<Target>
  <LocURI>./mail/bruce1?SUMMARY&EQ;Project%20XXX%20Review&OR;DESCRIPTION&E
Q;Project%20XXX%20Review</LocURI>
</Target>
```

4.18.5 Email Media Object Filter

The filter for contact media objects makes use of the MIME email CGI script tags conformant to the following ABNF.

```
Email-script = email-filter-item *(log-sep email-filter-item)

email-filter-item = email-tag log-op vcard-value

email-tag = "FROM" / "TO" / "SUBJECT" / "MID" / "CID" / "MARK"
          / exp-tag

email-value = string-value       ;MUST be a valid value for the MIME header
                                 ;or SyncML Meta element type value
```

The values of "FROM", "TO" and "REPLYTO" are any valid RFC822 email address string. The values of "SUBJECT" are any string of case sensitive, visible UTF-8 characters. The values of "MID" and "CID" are valid message identifiers. The email tag "MARK" is an element type used in the SyncML Meta element type content. The valid values include "READ", "UNREAD". Other values can also be specified for this tag. If CGI scripting is supported for this object type, then the "FROM", "TO" and "SUBJECT" CGI scripting primitives MUST be supported.

The following is an example of a value for a Target LocURI element type that filters the synchronization of the email target database to all those records that are unread and have a subject that contains the string "Project XXX". The search values for "SUBJECT" are case sensitive.

```
<Target>
  <LocURI>./mail/bruce9?MARK&EQ;UNREAD&AND;SUBJECT&EQ;Project%20XXX</LocUR
I>
</Target>
```

5 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, content model, and examples of use.

Examples in this section make use of XML snippets. They are not intended to be complete XML documents. They are only provided to illustrate an example usage of the element type in question.

Reasons for not using XML attribute lists in the SyncML representation protocol include simplicity of design and because of ongoing confusion over when and how to use element types versus attributes lists.

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.

5.1 Common Use Elements

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

5.1.1 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`

Restrictions: If the element type is present in a Delete command that contains a sequence of `Item` element types, then it applies to all of the data items.

If this element type is not specified, then the deleted data need not be archived by the recipient prior to being deleted.

If the recipient does not support this function then the response status code `501` (Not Implemented) MUST be returned.

Content Model:

```
(EMPTY)
```

Attributes: None.

Example:

```
<Delete>
  <CmdID>1234</CmdID>
  <Archive/>
  <Item>
    <Target><LocURI>./11</LocURI></Target>
  </Item>
</Delete>
```

5.1.2 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 4.13, "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 synchronization session. The session is authenticated.

Content Model:

```
(Meta)
```

Attributes: None.

Example: The following is an example of a SyncML "Basic" authentication challenge. The password and userid are requested to be Base64 character encoded. The type and format of the authentication scheme are specified by the meta-information in the `Meta` element type.

```
Status>
  <MsgRef>0<MsgRef>
  <Cmd>SyncHdr</Cmd>
  <TargetRef>http://www.datasync.org/servlet/syncit</TargetRef>
  <SourceRef>IMEI:001004FF1234567</SourceRef>
  <Chal>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
  </Chal>
  <Data>401</Data>
</Status>
```

The following is an example of a SyncML "MD5 digest" authentication challenge. The MD5 Digest is requested to be Base64 character encoded. The type and format of the authentication scheme, as well as the next nonce are specified by the meta-information in the `Meta` element type.

```
Status>
  <MsgRef>0<MsgRef>
  <Cmd>SyncHdr</Cmd>
  <TargetRef>http://www.datasync.org/servlet/syncit</TargetRef>
  <SourceRef>IMEI:001004FF1234567</SourceRef>
```

```
  <Chal>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
      <NextNonce xmlns='syncml:metinf'>ZG9iZWhhdmUNCg==</NextNonce>
    </Meta>
   </Chal>
  <Data>401</Data>
</Status><Chal>
```

5.1.3 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.

Example:

```
<Status>
  <CmdID>4321</CmdID>
  <MsgRef>1<MsgRef>
  <CmdRef>1234<CmdRef>
  <Cmd>Add</Cmd>
  <TargetRef>./mail/bruce1</TargetRef>
  <Data>401</Data>
  <Item>
    <Meta>
      <NextNonce xmlns='syncml:metinf'>F00BAC==</NextNonce>
    </Meta>
  </Item>
</Status>
```

5.1.4 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.

Example:

```
<Add>
  <CmdID>1234<CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Source><LocURI>./12</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>text/directory;profile=vCard</Type>
    </Meta>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Smith;Bruce
N:Bruce Smith
TEL;TYPE=WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
</Add>
```

5.1.5 CmdRef

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

Parent Elements: `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.

Example:

```
<Status>
  <CmdID>4321</CmdID>
  <MsgRef>1<MsgRef>
  <CmdRef>1234<CmdRef>
  <Cmd>Add</Cmd>
  <TargetRef>./mail/bruce1</TargetRef>
```

```
  <Data>401</Data>
  <Item>
    <Meta>
    <NextNonce xmlns='syncml:metinf'>ZG9iZWhhdmUNCg==</NextNonce>
    </Meta>
  </Item>
</Status>
```

5.1.6 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. The `Data` element type specifies the credential value. The types for these SyncML authentication schemes are specified in Section 4.13, "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 apply to it.

Content Model:

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

Attributes: None.

Example: The following is an example of a MD5 digest authentication credential scheme consisting of the character string `bruce1:ohbehave:nonce`. The MD5 Digest is also Base64 character encoded. The type and format of the credential, as well as the next nonce are specified by the meta-information in the `Meta` element type.

```
<Cred>
  <Meta>
    <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
    <Format xmlns='syncml:metinf'>b64</Format>
  </Meta>
  <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
```

```
</Cred>
```

5.1.7 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 [11] specifies the semantics of the different SyncML packages for the SyncML synchronization protocol.

Content Model:

```
(EMPTY)
```

Attributes: None.

Example:

```
<SyncML>
  <SyncHdr>...blah, blah...</SyncHdr>
  </SyncBody>
    ...blah, blah...
    </Final>
  </SyncBody>
</SyncML
```

5.1.8 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`

Restrictions: The value of the element type MUST BE a valid [6] formatted International language identifier. The semantics for this element type are quite different than the `xml:lang` attribute defined in [14], which indicates the International language identifier for the content information of any element type. This element type MUST NOT be used to specify the actual language of element types in a SyncML document. The `xml:lang` attribute MUST BE used for this latter purpose.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Get>
```

```
  <CmdID>5</CmdID>
  <Lang>en-US</Lang>
  <Item>
    <Target><LocURI>./F123456F</LocURI></Target>
    <Source><LocURI>./10</LocURI></Target>
  </Item>
</Get>
```

5.1.9 LocName

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

Parent Elements: `Target`, `Source`

Restrictions: A target- or source-specific display name to be associated with the associated `LocURI` value in the `Target` or `Source` element type.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Item>
  <Target>
    <LocURI>./mail/bruce1</LocURI>
    <LocName>Bruce Smith's InBox</LocName></Target>
</Item>
```

5.1.10 LocURI

Usage: Specifies a 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. Section 4.17, "Target and Source Addressing" provide restrictions on the values for the `LocURI` element type.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.datasync.org/servlet/syncit/</LocURI></Target
>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>
```

5.1.11 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

Example:

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.syncml.host.com/</LocURI></Target>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>
```

5.1.12 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

Example:

```
<Status>
  <CmdID>4321</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>1234</CmdRef>
  <Cmd>Add</Cmd>
  <Data>200</Data>
</Status>
```

5.1.13 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`

Restrictions: When specified, the recipient MUST NOT return a `Status` command for the associated SyncML command. If specified on the SyncHdr element type, the recipient MUST NOT return any `Status` for any of the commands in the current SyncML message.

Content Model:

```
(EMPTY)
```

Attributes: None

Example:

```
<Replace>
  <CmdID>1</CmdID>
  <NoResp/>
  <Item>
    <Source><LocURI>./127</LocID></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>text/directory profile=vCard</Type>
    </Meta>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith;Bruce
TEL;TYPE=WORK;VOICE;MSG:+1-919-555-9999</Data>
ADR:;;123 Main St.;Anywhere;CA;;US
END:VCARD
    </Data>
  </Item>
</Replace>
```

5.1.14 NoResults

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

Parent Elements: `Get`, `Search`

Restrictions: The indicator is used to force the results of a `Get` or `Search` command to remain on the recipient system in a temporary repository specified by the Source element type within the command.

If the recipient does not support local temporary repositories, the (`406`) `Optional feature not supported` exception conciliation will be created.

Content Model:

```
(EMPTY)
```

Attributes: None

Example:

```
<Search>
  <CmdID>3</CmdID>
  <NoResults/>
  <Source><LocURI>./bruce1/emp_tabl.db</LocURI></Source>
  <Meta><Type xmlns='syncml:metinf'>application/sql</Type></Meta>
  <Data>SELECT EQ *
WHERE "FN" EQ "Bruce Smith"
</Data>
</Search>
```

5.1.15 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.

CGI scripting can be used to convey "hints" to the recipient about when a response should be attempted. The script parameter "after" is used for this purpose. The value of the parameter is an UTC based, ISO 8601 basic format, complete representation for a date and time of day value (e.g., 20000502T224952Z for May 02, 2000 at 22 hours, 49 minutes and 52 seconds UTC). Additional script parameter can be prefixed or appended to this SyncML specific CGI script parameter. See Section 4.18.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: In the following example, a response URI is specified with the application specific CGI script parameter of `user`, which is prefixed to the SyncML script parameter of `after`.

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target>
    <LocURI>IMEI:001004FF1234567</LocURI>
    <LocName>Bruce's Mobile Device</LocName><Target>
  <Source>
    <LocURI>http://www.datasync.org/servlet/syncit/bruce1</LocURI>
  </Source>
  <RespURI>http://www.datasync.org/servlet/syncit/bruce1?user=jsmith&after
=20000512T133000Z</RespURI>
</SyncHdr>
```

5.1.16 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.

Content Model:

```
(#PCDATA)
```

Attributes: None

Example:

```
<SyncML>
  <SyncHdr>
    <VerDTD>1.0</VerDTD>
    <VerProto>SyncML/1.0</VerProto>
    <SessionID>1</SessionID>
    <MsgID>3</MsgID>
    <Target>
      <LocURI>IMEI:001004FF1234567</LocUR>
    </Target>
    <Source>
      <LocURI>http://www.datasync.org/servlet/syncit/</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
  ...blah, blah...
  </SyncBody>
<SyncML>
```

5.1.17 SftDel

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

Parent Elements: `Delete`

Restrictions: The data item is deleted from the data store but not from the set of synchronization data. The "Soft Delete" can be specified by a SyncML server to free up storage resources in the SyncML client prior to a synchronization operation. If not present, then the semantics of the `Delete` command are a "Hard Delete" of the data item. In addition, the SyncML client can specify the "Soft Delete" to free up storage resources in the SyncML client prior to a synchronization operation with the SyncML server.

The SyncML client MUST maintain the LUID (Local Unique Identifier) associated with the soft-deleted item that server(s) can re-use the LUID if the item is modified by a server.

The SyncML server MUST not delete the map items associated with the "Soft Deleted" items.

If the SyncML client does not support the "Soft Delete", then, a `(406) Optional feature not supported` MUST be returned in the Status command.

In a two-way synchronization, if the SyncML client specifies a "Soft Delete" for an item that has already been "Hard Deleted" on the SyncML server, then a (`423`) `Soft-delete conflict` MUST be returned in the Status command.

Content Model:

```
EMPTY
```

Attributes: None

Example:

```
<Delete>
  <CmdID>3456</CmdID>
  <SftDel/>
  <Item>
    <Target><LocURI>./11<LocURI></Target>
  </Item>
</Delete>
```

5.1.18 Source

Usage: Specifies source routing or mapping information.

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

Restrictions: Section 4.17, "Target and Source Addressing" provides semantics on the content of the `Source` element type.

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 `Map` element type, the `Source` element type specifies the routing information of the database that originated the map definition. When specified in the `MapItem` element type, the `Source` element type specifies the identifier of the client item.

When specified in the `Search` element type, the `Source` element type specifies the source routing information of the database that the search command is to be executed against.

When specified in the `Sync` element type, the `Source` element type specifies the source routing information of the database originating the data synchronization request.

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.

Example: The following is an example of the usage in a SyncHdr element type.

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>3</MsgID>
  <Target><LocURI>http://www.syncml.org/servlet/syncit/</LocURI></Target>
  <Source>
    <LocURI>IMEI:001004FF1234567</LocURI>
    <LocName>Bruce's Mobile Device</LocName>
  </Source>
</SyncHdr>
```

The following is an example of the usage in an Item element type.

```
<Replace>
  <CmdID>4567</CmdID>
  <Item>
    <Target><LocURI>./bruce1/pnab</LocURI></Target>
    <Source><LocURI>./contacts</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>text/directory profile=vCard</Type>
    </Meta>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Bruce Smith
N:Smith;Bruce
TEL;TYPE=WORK;VOICE;MSG:+1-919-555-9999
END:VCARD</Data>
  </Item>
</Replace>
```

The following is an example of the usage in a Map and MapItem element type.

```
<Map>
  <CmdID>3456</CmdID>
  <Target>
    <LocURI>http://www.datasync.org/servlet/syncit?USER=jsmith&db=Employee
_Table.db</LocURI>
    </Target>
  <Source><LocURI>./tables</LocURI></Source>
  <MapItem>
    <Target><LocURI>./0123456789ABCDEF</LocURI></Target>
    <Source><LocURI>./12</LocURI></Source>
  </MapItem>
</Map>
```

5.1.19 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.

Example:

```
<Status>
  <CmdID>4321</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>1234</CmdRef>
  <Cmd>Add</Cmd>
  <TargetRef>./01234567890ABCDEF</TargetRef>
  <SourceRef>./12</SourceRef>
  <Data>200</Data>
</Status>
```

5.1.20 Target

Usage: Specifies target routing or mapping information.

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

Restrictions: Section 4.17, "Target and Source Addressing" provides semantics on the content of the `Target` element type.

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 `Map` element type, the `Target` element type specifies the routing information of the database that is to maintain the map definition. When specified in the `MapItem` element type, the `Target` element type specifies the identifier of the server item.

When specified in the `Search` element type, the `Target` element type specifies the target routing information where the search results are to be temporarily stored. In this case, the results will be specified as the source for a subsequent SyncML command.

When specified in the `Sync` element type, the `Target` element type specifies the source routing information of the database receiving the data synchronization request.

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.

Example: The following is an example of the usage in a SyncHdr element type.

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>3</MsgID>
  <Target><LocURI>http://www.syncml.org/servlet/syncit/</LocURI></Target>
  <Source>
    <LocURI>IMEI:001004FF1234567</LocURI>
    <LocName>Bruce's Mobile Device</LocName>
  </Source>
</SyncHdr>
```

The following is an example of the usage in an Item element type.

```
<Replace>
  <CmdID>2</CmdID>
  <Item>
    <Target><LocURI>./bruce1/pnab</LocURI></Target>
    <Source><LocURI>./contacts</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>text/directory profile=vCard</Type>
    </Meta>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Bruce Smith
N:Smith;Bruce
TEL;TYPE=WORK;VOICE;MSG:+1-919-555-9999
END:VCARD</Data>
  </Item>
</Replace>
```

The following is an example of the usage in a Map and MapItem element type.

```
<Map>
  <CmdID>3456</CmdID>
  <Target>
    <LocURI>http://www.datasync.org/servlet/syncit?USER=jsmith&db=Employee
_Table.db</LocURI>
    </Target>
  <Source><LocURI>./tables</LocURI></Source>
  <MapItem>
    <Target><LocURI>./0123456789ABCDEF</LocURI></Target>
    <Source><LocURI>./12</LocURI></Source>
  </MapItem>
</Map>
```

5.1.21 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.

Example:

```
<Status>
  <CmdID>4321</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>1234</CmdRef>
  <Cmd>Add</Cmd>
  <TargetRef>./01234567890ABCDEF</TargetRef>
  <SourceRef>./12</SourceRef>
  <Data>200</Data>
</Status>
```

5.1.22 VerDTD

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

Parent Elements: `SyncHdr`

Restrictions: Major reversions 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 `1.0`. The element type MUST be included in the SyncHdr.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target>
    <LocURI>IMEI:001004FF1234567</LocURI><LocName>Bruce's Mobile
Phone</LocName>
  </Target>
```

```
  <Source>
      <LocURI>http://www.datasync.org/servlet/syncit/bruce1</LocURI>
  </Source>
</SyncHdr>
```

5.1.23 VerProto

Usage: Specifies the major and minor version identifier of the SyncML synchronization protocol specification used with the SyncML Message.

Parent Elements: `SyncHdr`

Restrictions: Major reversions of the specification create incompatible changes that will require a new SyncML synchronization engine. Minor revisions involve changes that do not impact basic compatibility of the synchronization engine. When the SyncML workflow conforms to this revision of the SyncML synchronization protocol specification the value MUST be `1.0`. 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.

Example:

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target>
    <LocURI>IMEI:001004FF1234567</LocURI><LocName>Bruce's Mobile
Phone</LocName>
  </Target>
  <Source>
      <LocURI>http://www.datasync.org/servlet/syncit/bruce1</LocURI>
  </Source>
</SyncHdr>
```

5.2 Message Container Elements

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

5.2.1 SyncML

Usage: Specifies the container for a SyncML Message.

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

Restrictions: Within transports that support MIME content-type identification, this object MUST BE identified as **`application/vnd.syncml+xml` (for clear-text, XML representation) or `application/vnd-syncml+wbxml` (for binary, WBXML representation)**.

Content Model:

```
(SyncHdr, SyncBody)
```

Attributes:

Name	Type	Occurrance	Description
xmlns	CDATA	REQUIRED	Value must be the text: 'SYNCML:SYNCML1.0'

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.0'>
  <SyncHdr>
    <VerDTD>1.0</Version>
    <VerProto>SyncML/1.0</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.datasync.org/servlet/syncit</LocURI></Target>
    <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
  </SyncHdr>
  <SyncBody>
    ...blah, blah...
  </SyncBody>
</SyncML>
```

5.2.2 SyncHdr

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

Parent Elements: `SyncML`

Restrictions: If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for any of the commands in the current SyncML message.

If specified, the optional `NoResp` element type indicates the recipient MUST NOT return any `Status` commands for any of the commands in the current SyncML message.

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

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.0'>
  <SyncHdr>
    <VerDTD>1.0</Version>
    <VerProto>SyncML/1.0</VerProto>
    <SessionID>1</SessionID>
```

```
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.datasync.org/servlet/syncit</LocURI></Target>
    <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
  </SyncHdr>
  <SyncBody>
    ...blah, blah...
  </SyncBody>
</SyncML>
```

5.2.3 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)+, Final?)
```

Attributes: None.

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.0'>
  <SyncHdr>
    <VerDTD>1.0</Version>
    <VerProto>SyncML/1.0</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target><LocURI>IMEI:001004FF1234567</LocURI></Target>
    <Source>
      <LocURI>http://www.datasync.org/servlet/syncit</LocURI>
    </Source>
    <Cred>
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
    </Cred>
  </SyncHdr>
  <SyncBody>
    <Get>
      <CmdID>1234</CmdID>
      <Item>
        <Target><LocURI>./devinf10</LocURI></Target>
      </Item>
    </Get>
  </SyncBody>
</SyncML>
```

5.3 Data Description Elements

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

5.3.1 Data

Usage: Specifies discrete SyncML data.

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

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 either 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.

Example: The following is an example of an Item with data that does not contain any mark-up.

```
<Item>
  <Data>John Smith, +1-919-555-1234</Data>
</Item>
```

The following is an example of a meta-information mark-up data.

```
<Meta>
  <Format xmlns='syncml:metinf'>xml</Format>
  <Type xmlns='syncml:metinf'>application/vnd.syncml-devinf+xml</Type>
  <Data>
    <DevInf xmlns='syncml:devinf'>
      <Man xmlns='syncml:devinf'>IBM</Man>
      <Mod xmlns='syncml:devinf'>WorkPad<Mod>
      <DevTyp xmlns='syncml:devinf'>pda</DevTyp>
      <DevID xmlns='syncml:devinf'>J. Smith</DevID>
      <FwV xmlns='syncml:devinf'>PalmOSv3.0</FwV>
      <OEM xmlns='syncml:devinf'>Palm, Inc.</OEM>
    </DevInf>
  </Data>
</Meta>
```

5.3.2 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.

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.

Example:

```
<Add>
  <CmdID>1<CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Source><LocURI>./15</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:devinf'>text/directory profile=vCard</Type>
    </Meta>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Smith;Bruce
N:Bruce Smith
TEL;TYPE=WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
</Add>
```

5.3.3 Meta

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

Parent Elements: `Add`, `Atomic`, `Chal`, `Copy`, `Cred`, `Delete`, `Get`, `Item`, `Map`, `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 `Map`, then the scope of the meta-information includes all the contained `MapItem` element types.

When specified in the `Search`, the element type specifies the search grammar to be used.

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

Example:

```
<Meta>
  <Format xmlns='syncml:metinf'>xml</Format>
  <Type xmlns='syncml:metinf'>application/vnd.syncml-devinf+xml</Type>
  <Data>
    <DevInf xmlns='syncml:devinf'>
      <Man xmlns='syncml:devinf'>IBM</Man>
      <Mod xmlns='syncml:devinf'>WorkPad<Mod>
      <DevTyp xmlns='syncml:devinf'>pda</DevTyp>
      <DevID xmlns='syncml:devinf'>J. Smith</DevID>
      <FwV xmlns='syncml:devinf'>PalmOSv3.0</FwV>
      <OEM xmlns='syncml:devinf'>Palm, Inc.</OEM>
    </DevInf>
  </Data>
</Meta>
```

5.4 Protocol Management Elements

5.4.1 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 the associated command specified multiple `Item` element types, then the corresponding `Status` can apply to the status of one, all or some of the `Item` element types. In these latter cases, the scope of the `Status` is determined by the `SourceRef` element type values specified in the `Status` command (i.e., which of the `Item` element types the `Status` applies to).

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.

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. If the `MsgRef` is not present in a `Status` element type, then the `MsgRef` value of "1" MUST be assumed.

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".

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 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 in the. 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. 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
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	Unauthorized. The requested command failed because proper authentication MUST be provided by the originator. If no authentication was provided in the request, a suitable challenge MAY also be returned in the Item element type in the Status. 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 doesn't 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	Authentication required. 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 Item 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.
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 ReplyURI 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 the 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 ReplyURI 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 synch with the server.
509	Authentication required. The response code indicates that the server MUST first authenticate with the originator. The client SHOULD also return the suitable challenge in the Item element type in the Status.
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.

Content Model:

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

```
Item*)
```

Attributes: None.

Example:

```
<SyncBody>
  <Status>
    <CmdID>8765</CmdID>
    <MsgRef>1<MsgRef>
    <CmdRef>1234<CmdRef>
    <Cmd>Add</Cmd>
    <TargetRef>./bruce1</TargetRef>
    <SourceRef>IMEI:001004FF1234567</SourceRef>
    <Data>401</Data>
  </Status>
</SyncBody>
```

5.5 Protocol Command Elements

5.5.1 Add

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

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

Restrictions: The `Add` command is generally used to convey to the recipient any additions made to the originator's database. For example, a mobile device will indicate to the network server any additions made to the local calendar database.

The originator of the command SHOULD determine what features/properties of the data item are supported by the recipient and only send supported properties. The device information document on the recipient can contain this information.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the parent `Sync` element type. If not specified there, the default authentication credential is take from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data item added to the database. The `Target` and `Source` specified within the Item

element type SHOULD be a relative URI, as relative to the corresponding `Target` and `Source` specified in the parent `Atomic`, `Sequence` or `Sync` command.

The recipient MAY assign new local identifiers for the data items specified in this command. However, in such cases the recipient MUST also notify the originator of the item identifier correlation by returning a `Map` command.

If the command completed successfully, then the (`201`) `Item added` exception condition is created by the command.

If the recipient determines that the data item already exists on the recipient's database, then the (`418`) `Already exists` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

If there is insufficient space on the recipient database for the data item, then the (`420`) `Device full` exception condition is created by the command.

If the media type or format for the data item is not supported by the recipient, then the (`415`) `Unsupported media type or format` exception condition is created by the command.

Content Model:

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

Attributes: None.

Example:

```
<Add>
  <CmdID>12345</CmdID>
  <  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Meta>
      <Format xmlns='syncml:metinf'>chr</Format>
      <Type xmlns='syncml:metinf'>text/x-vcard</Type>
  </Meta>
  <Item>
    <Source>
      <LocURI>./2</LocURI>
    </Source>
```

```
      <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith;Bruce
TEL;WORK;VOICE:+1-919-555-1234
TEL;WORK;FAX:+1-919-676-9876
EMAIL;INTERNET:bruce1@host.com
END:VCARD
      </Data>
  </Item>
</Add>
```

5.5.2 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: `SyncBody`

Restrictions: The `Alert` command is specifically used to convey notifications, such as data synchronization requests, to the recipient. For example, a mobile device will use this command to initiate a "client-initiated, two-way synchronization" with a network server.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

The optional `Data` element type specifies the type of alert.

Optionally, one or more `Item` element types can be specified. The `Item` element type specifies parameters for the `Alert` command. The `Target` and `Source` specified within the `Item` element type MUST be an absolute URI.

If the command and the associated `Alert` action are completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the command was accepted successfully, but the `Alert` action has not yet been executed successfully, then the (`202`) `Accepted for processing` exception condition is created by the command. A subsequent exception condition can be created to relate the eventual completion status of the associated `Alert` action.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) Unauthorized exception condition is created by the command. If no authentication credentials were specified, then (407) Authentication required exception condition is created by the command. A suitable challenge can also be returned.

If the data synchronization protocol does not allow the Alert command to be specified at within the current SyncML package (i.e., the Alert was issued in an incorrect protocol sequence), then the command creates the (405) Command not allowed exception condition.

If the specified Alert command is not supported by the recipient, the (406) Optional feature not supported exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (500) Command failed exception condition.

If the Alert command didn't include all the correct parameters in the Item element type, then the (412) Incomplete command exception condition is created by the command.

If the media type or format for the data item is not supported by the recipient, then the (415) Unsupported media type or format exception condition is created by the command.

Alert Code Value	Name	Description
Alert Codes used for user alerts		
100	DISPLAY	Show. The Data element type contains content information that should be processed and displayed through the user agent.
101-150	-	Reserved for future SyncML usage.
Alert Codes used at the synchronization initialization		
200	TWO-WAY	Specifies a client-initiated, two-way sync.
201	SLOW SYNC	Specifies a client-initiated, two-way slow-sync.
202	ONE-WAY FROM CLIENT	Specifies the client-initiated, one-way only sync from the client to the server.
203	REFRESH FROM CLIENT	Specifies the client-initiated, refresh operation for the one-way only sync from the client to the server.
204	ONE-WAY FROM SERVER	Specifies the client-initiated, one-way only sync from the server to the client.
205	REFRESH FROM SERVER	Specifies the client-initiated, refresh operation of the one-way only sync from the server to the client.
Alert Codes used by the server when alerting the sync.		
206	TWO-WAY BY SERVER	Specifies a server-initiated, two-way sync.

207	ONE-WAY FROM CLIENT BY SERVER	Specifies the server-initiated, one-way only sync from the client to the server.
208	REFRESH FROM CLIENT BY SERVER	Specifies the server-initiated, refresh operation for the one-way only sync from the client to the server.
209	ONE-WAY FROM SERVER BY SERVER	Specifies the server-initiated, one-way only sync from the server to the client.
210	REFRESH FROM SERVER BY SERVER	Specifies the server-initiated, refresh operation of the one-way only sync from the server to the client.
211-220	-	Reserved for future SyncML usage.
Special Alert Codes		
221	RESULT ALERT	Specifies a request for sync results.
222	NEXT MESSAGE	Specifies a request for the next message in the package.
223-250	-	Reserved for future SyncML usage.

Content Model:

```
(CmdID, NoResp?, Cred?, Data?, Item+)
```

Attributes: None.

Example: The following example is an `Alert` type `100` that displays a message on the recipient's console or display.

```
<Alert>
  <CmdID>3456</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Data>100</Data>
  <Item>
    <Data>You have new mail!</Data>
  </Item>
</Alert>
```

The following example is for a hypothetical `Alert` type `299` that notifies the recipient of new mail items.

```
<Alert>
  <CmdID>5678</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
```

```
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Data>299</Data>
  <Item>
    <Data><Source><LocURI>mid:msg1@host.com</LocURI></Source></Data>
  </Item>
  <Item>
    <Data><Source><LocURI>mid:msg2@host.com</LocURI></Source></Data>
  </Item>
  <Item>
    <Data><Source><LocURI>mid:msg3@host.com</LocURI></Source></Data>
  </Item>
</Alert>
```

5.5.3 Atomic

Usage: Specifies the SyncML command to request that the subordinate executed as a set.

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

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

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 remainder of the command consists of one or more `Add`, `Delete`, `Copy`, `Atomic`, `Map`, `Replace`, `Sequence` or `Sync` SyncML commands that are the scope of the `Atomic` functionality.

If the command and the associated individual commands are completed successfully, then the (`200`) `OK` exception condition is created by the command.

If an error occurs while performing an individual command specified in an Atomic element type, then the (`507`) `Atomic failed` exception condition is created by the command. The error status code indicates the failure of the complete `Atomic` command. Separate, individual error status code can also be created that identify specific errors that created the failure.

Content Model:

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

Attributes: None.

Example:

```
<Atomic>
  <CmdID>1234</CmdID>
  <Add>
    <CmdID>1235</CmdID
    <Cred>
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
    </Cred>
    <Item>
      <Target><LocURI>./devinf10/pen</LocURI></Target>
      <Data>Yes</Data>
    </Item>
  </Add>
  <Replace>
    <CmdID>12346</CmdID>
    <Cred>
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
    </Cred>
    <Item>
      <Target><LocURI>./devinf10/version</LocURI></Target>
    <Data>20000401T133000Z</Data>
  </Replace>
</Atomic>
```

5.5.4 Copy

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

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

Restrictions: It is implementation dependent whether a physical copy of the item is made in the recipient, or whether a shortcut or pointer is created to the source item in the target location.

The `Copy` command in this version of the specification is NOT intended to be used to attempt to change the media type of a data item, compress the data item or otherwise transform a target data item.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the parent `Sync` element type. If not specified there, the default authentication credential is take from the

`SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data item to be copied on the recipient's database. If specified within a `Sync` element type, the `Target` and `Source` specified within the `Item` element type in the `Copy` command SHOULD be a relative URI, as relative to the corresponding `Target` and `Source` specified in the parent `Sync` command. If specified within a `Sequence` or `SyncBody` element type, the `Target` and `Source` specified within the `Item` element type in the `Copy` command SHOULD be an absolute URI.

The recipient MAY assign new local identifiers for the data items specified in this command. However, in such cases the recipient MUST also notify the originator of the item identifier correlation by returning a `Map` command.

If the command completed successfully, then the (`201`) `Item added` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the target data item already exists in the recipient database, then the (`418`) `Already exists` exception condition is created by the command.

If there is insufficient space in the recipient database for the data item, then the (`420`) `Device full` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

If the an error occurs while the recipient copying the data item within the recipient's data base, then the (`510`) `Data store failure` exception condition is created by the command.

Content Model:

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

Attributes: None.

Example:

```
<Copy>
```

```
  <CmdID>12345</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Target><LocURI>mid:msg1@host.com</LocURI></Target>
    <Source><LocURI>./mail/bruce1/folders/Project%20XYZ</LocURI></Source>
  </Item>
  <Item>
    <Target><LocURI>mid:msg2@host.com</LocURI></Target>
    <Source><LocURI>./mail/bruce1/folders/Admin</LocURI></Source>
  </Item>
</Copy>
```

5.5.5 Delete

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

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

Restrictions: The `Delete` command is generally used to permanently erase data items from the recipient's database. However, the command can also be used to temporarily remove data items from the recipient's database in order to create room for a subsequent `Add` command. This is termed a "soft delete".

Implementations that support both a preliminary, "Mark for Deletion" and a physical "Delete" capabilities, MUST use this command for the latter, "Delete" capability. The preliminary "Mark for Deletion" MUST be specified using the `Mark` meta-information in a `Replace` command.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

If specified, the optional `Archive` element type indicates that the recipient SHOULD preserve a copy of the data prior to deleting it from the database.

If the recipient implementation doesn't support archiving of data (i.e., doesn't support the `Archive` feature), then the command will create the (`210`) `Delete without archive` exception condition. However, under this condition, the data has been successfully deleted from the recipient database.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the parent `Sync` element type. If not specified there, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data item deleted from the database. The `Target` and `Source` specified within the Item element type SHOULD be a relative URI, as relative to the corresponding `Target` and `Source` specified in the parent `Atomic`, `Sequence` or `Sync` command.

In applications (e.g., email) where the "delete" concept involves "moving" a data item from one folder to a special "Deleted" folder, this is achieved in SyncML by the sequence of two SyncML functions; one, the `Add` of the specified data item to the "Deleted" folder and two, the subsequent `Delete` of the corresponding item from the original folder.

The recipient of a Delete command can delete any subset of the specified data elements. However, if all of the requested data was not deleted, then the (`206`) `Partial content` exception condition is created by the command. If a `Status` command is returned for this exception condition, then the identifiers of the data items not deleted SHOULD be returned also.

If the recipient determines that the data item doesn't exists on the recipient's database, then the (`404`) `Not found` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

In synchronization protocol cases where the client sends a `Map` command to the server, the server MUST always specify the client identifier for any data items to be deleted. Otherwise, the (`412`) `Incomplete command` exception condition is created and no data items will be deleted by the client.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

Content Model:

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

Attributes: None.

Example: The following is an example to delete a data item but archive it first.

```
<Delete>
  <CmdID>12345</CmdID>
  <Archive/>
  <Cred>
    <Meta>
```

```
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Source><LocURI>./4</LocURI></Source>
  </Item>
</Delete>
```

The following is an example to "soft delete" a number of data items to allow room on the device for a subsequent `Add` or `Copy` command, not specified in this example.

```
<Delete>
  <CmdID>12345</CmdID>
  <SftDel/>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Target><LocURI>./5</LocURI></Target>
  </Item>
  <Item>
    <Target><LocURI>./10</LocURI></Target>
  </Item>
  <Item>
    <Target><LocURI>./8</LocURI></Target>
  </Item>
</Delete>
```

5.5.6 Exec

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

Parent Elements: `SyncBody`

Restrictions: The Exec command is used to perform remote procedure calls on a remote network device. This command permits a mobile device to invoke network applications that can create updates to databases that the mobile device wishes to synchronization with. Likewise, the command can be used by network servers to invoke local mobile device applications.

Both implementors and users SHOULD take appropriate steps to avoid security threats introduced by a remote procedure call mechanism, such as the `Exec` command, in data synchronization products.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in this other element type, then the command is specified without an authentication credential.

One `Item` element type MUST be specified. The `Item` element type specifies the target address and parameters for the remote procedure call. The `Target` specified within the Item element type MUST be an absolute URI.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the recipient successfully accepts the command and has invoked the remote procedure call but the remote procedure has not yet successfully completed, then the (`202`) `Accepted for processing` exception condition is created by the command.

If an entity other than the remote procedure returns request status codes to the originator, then the (`203`) `Non-authoritative response` exception condition is created by the command. For example, an ERP or workflow procedure call can result in a subsystem daemon or process returning responses to the originator.

If the targeted remote procedure has been permanently relocated, then the (`301`) `Moved permanently` exception condition is created by the command.

If the targeted remote procedure has been temporarily moved, then the (`302`) `Found` exception condition is created by the command.

If the targeted remote procedure must be executed through a proxy, then the (`305`) `Use proxy` exception condition is created by the command.

If the syntax of the `Exec` command was not specified correctly, then the (`400`) `Bad request` exception condition is created by the command.

If the originator doesn't have sufficient rights to issue an `Exec` command on the recipient, then the (`403`) `Forbidden` exception condition is created by the command. For this exception condition, there does not exist any authentication credential for the originator with sufficient rights. The exception condition (`404`) `Not found` can alternately be specified when the recipient doesn't want to make public why the request was denied.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the `Exec` command is not allowed to be invoked on the recipient, then the (`403`) `Forbidden` exception condition is created by the command.

If the specified `Exec` command cannot be found on the recipient, then the (`404`) `Not found` exception condition is created by the command.

If the specified remote procedure call no longer exists on the recipient, then the (`410`) `Gone` exception condition is created by the command.

If the media type or format for the remote procedure call specified in the `Item` is not supported by the recipient, then the (`415`) `Unsupported media type or format` exception condition is created by the command.

If the specified remote procedure is currently busy and a retry later is possible, then the (`417`) `Retry later` exception condition is created by the command. The date/time after which the originator should retry the command SHOULD also be returned.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

If there is any error in the remote execution of the command, then the (`506`) `Processing error` exception condition is created by the command.

Content Model:

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

Attributes: None.

Example: The following is an example of a hypothetical employee-change-request process being invoked by the `Exec` command.

```
<Exec>
  <CmdID>1234</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Target>
      <LocURI>http://www.datasync.org/cgi?proc=ecrproc.exe</LocURI>
    </Target>
    <Data><ec:ecr xmlns:ec='abccorp:ecapp'>
<ec:ecrcmd>PROMOTION</ec:ecrcmd>
<ec:oldbnd>10</ec:oldbnd><ec:newbnd>D</ec:newbnd>
<ec:name>Bruce Smith</ec:name>
<ec:enum>L01234</ec:enum><ec:dop>20000523</ec:dop></ec:ecr></Data>
  <Item>
</Exec>
```

5.5.7 Get

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

Parent Elements: `SyncBody`

Restrictions: There is no synchronization state information for data retrieved using the `Get` command.

Data returned from a `Get` command is returned in a `Results` element type in a subsequent SyncML message.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

If specified, the optional `Lang` element type specifies the language desired for any returned results.

The optional `Cred` element type specifies the authentication credential to be used for the command. If no present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in this other element type, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data items to be returned from the recipient. The `Target` and `Source` specified within the Item element type SHOULD be an absolute URI.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the command completed successfully but there is no content to return, then the (`204`) `No content` exception condition is created by the command.

If the command completed successfully but only a portion of the content is being returned, with the remainder being returned in subsequent `Results` commands, then the (`206`) `Partial content` exception condition is created by the command.

If the command specifies an ambiguous target with multiple matches, then the (`300`) `Multiple choices` exception condition is created by the command.

If the command must be issued through a proxy, then the (`305`) `Use proxy` exception condition is created by the command. The URI of the proxy SHOULD also be returned.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`)

`Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the specified data item doesn't exist on the recipient, then the (`404`) `Not found` exception condition is created by the command.

If the requested data item is too large to be transferred at this time, then the (`413`) `Request entity too large` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

If the media type or format for the data item is not supported by the recipient, then the (`415`) `Unsupported media type or format` exception condition is created by the command.

Content Model:

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

Attributes: None.

Example:

```
<Get>
  <CmdID>12345</CmdID>
   <Lang>en-US</Lang>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Target><LocURI>./telecom/pb<LocURI><Target>
    <Source><LocURI>http://www.datasync.com/servlet/</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>text/x-vCard</Type>
    </Meta>
  </Item>
</Get>
```

5.5.8 Map

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

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

Restrictions: The `Map` command specifies additions and deletions to the recipient's item identifier map table. Map tables are used to correlate small resolution item identifiers with larger resolution item identifiers. For example, if a mobile device has 2-byte item identifiers and a network server has 16-byte item identifiers, a map table is required to correlate an equivalent 2-byte and 16-byte identifier. Generally, map tables are maintained by the data synchronization engine on the network server. Item identifier map tables are not necessary

when the originator and the recipient databases are exact replicas of each other (i.e., the databases have the same physical schema).

If an item identifier map table is needed, it is the responsibility of the recipient maintaining the map table to perform item identifier translations when communicating synchronization commands with the mobile device necessitating the map table.

The `Map` command MUST be atomic, in nature. This means that the recipient MUST process either the entire list of mappings supplied, or none of them. If the operation fails, the recipient MUST specify an error status code in the requested response.

The `Map` command is "*item potent*", which means that if the recipient applies the same `Map` command more than once, the result must be the same as applying the `Map` command only once.

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

The `Target` and `Source` element types MUST be specified. The `Target` element type specifies the target address for the map table on the recipient. The `Source` element type specifies the source address for the map table on the originator.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

The optional `Meta` element type specifies meta-information defining the type of `Map` command.

One or more `MapItem` element types MUST be specified. The `MapItem` element type specifies an individual item identifier mapping.

There MUST only be a single exception condition associated with each `Map` command.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If there is insufficient space on the recipient for the map table items, then the (`420`) `Device full` exception condition is created by the command.

If the recipient encounters a data store failure while processing the command, then the (`510`) `Data store failure` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

Content Model:

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

Attributes: None.

Example: The following is an example of a `Map` command for creating three item identifier mappings.

```
<Map>
  <CmdID>1234</CmdID>
  <Target><LocURI>http://www.datasync.org/servlet/syncit</LocURI></Target>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <MapItem>
    <Target><LocURI>./0123456789ABCDEF</LocURI></Target>
    <Source><LocURI>./01</LocURI></Source>
  </MapItem>
  <MapItem>
    <Target><LocURI>./0123456789ABCDF0</LocURI></Target>
    <Source><LocURI>./02</LocURI></Source>
  </MapItem>
  <MapItem>
    <Target><LocURI>./0123456789ABCDF1</LocURI></Target>
    <Source><LocURI>./03</LocURI></Source>
  </MapItem>
</Map>
```

5.5.9 MapItem

Usage:

Parent Elements: `Map`

Restrictions: The `Source` element type specifies the relative URI for the source item identifier.

The `Target` element type specifies the relative URI for the target item identifier.

Content Model:

```
(Target, Source)
```

Attributes: None.

Example:

```
<MapItem>
  <Target><LocURI>./0123456789ABCDEF</LocURI></Target>
  <Source><LocURI>./01</LocURI></Source>
</MapItem>
```

5.5.10 Put

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

Parent Elements: `SyncBody`

Restrictions: There is no synchronization state information for data transferred using the `Put` command.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

If specified, the optional `Lang` element type specifies the language desired for any returned results.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in this other element type, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data items to be transferred to the recipient. The `Target` and `Source` specified within the Item element type SHOULD be an absolute URI.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the command must be issued through a proxy, then the (`305`) `Use proxy` exception condition is created by the command. The URI of the proxy SHOULD also be returned.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the Put command did not include the size of the data item to be transferred (i.e., in the Meta element type), then the (411) `Size required` exception condition is created by the command.

If the data item to be transferred is too large (e.g., there are restrictions on the size of data items transferred to the recipient), then the (413) `Request entity too large` exception condition is created by the command.

If the `Size` specified in the `Meta` element type was too large for the recipient (e.g., the recipient does not have sufficient input buffer for the data), then the (416) `Requested size too big` exception condition is created by the command.

If the media type or format for the data item is not supported by the recipient, then the (415) `Unsupported media type or format` exception condition is created by the command.

If the recipient device storage is full, then the (420) `Device full` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (500) `Command failed` exception condition.

Content Model:

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

Attributes: None.

Example: The following is an example of a Put command used to exchange device information.

```
<Put>
  <CmdID>12345</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Item>
    <Target>
      <LocURI>http://www.datasync.org/servlet/syncit/bruce1</LocURI>
    </Target>
    <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>application/xml</Type>
      <Format xmlns='syncml:metinf'>xml</Format>
    </Meta>
    <Data>
<DevInf xmlns='syncml:devinf'>
<Man>UltraLite Mobile, Ltd.</Man>
<FwV>3.0</FwV>
<FwD>19981015</FwD>
<DevID>001004FF1234567</DevID>
```

```
<Mem>
<TotalMaxMem>1046529</TotalMaxMem>
<TotalMaxID>1024</TotalMaxID>
</Mem>
</DevInf>
    </Data>
</Put>
```

5.5.11 Replace

Usage: Specifies the SyncML command to replace data.

Parent Elements: `Atomic, Sequence, Sync`

Restrictions: **Restrictions**: The `Replace` command is used to replace data on the recipient.

If the specified data item does not exist, then the command MUST be interpreted as an `Add` command.

The originator of the command SHOULD determine what features/properties of the data item are supported by the recipient and only send supported properties. The device information document on the recipient can contain this information.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If no present, the default authentication credential is taken from the parent `Sync` element type. If not specified there, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

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.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data item replaced in the database. The `Target` and `Source` specified within the Item element type SHOULD be a relative URI, as relative to the corresponding `Target` and `Source` specified in the parent `Atomic`, `Sequence` or `Sync` command.

The recipient MAY assign new local identifiers for the data items specified in this command. However, in such cases the recipient MUST also notify the originator of the item identifier correlation by returning a `Map` command.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command. However, if the command was interpreted as an `Add` command and the

command completed successfully, then the (201) `Item added` exception condition is created by this command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (407) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

Non-specific errors created by the recipient while attempting to complete the command create the (500) `Command failed` exception condition.

If there is insufficient space on the recipient database for updating the data item, then the (420) `Device full` exception condition is created by the command.

If the media type or format for the data item is not supported by the recipient, then the (415) `Unsupported media type or format` exception condition is created by the command.

Content Model:

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

Attributes: None.

Example: The following example specifies a source item that was replaced in the source database. The `Source` contains the relative URI of the item that was replaced. The absolute URI of the `Source` is specified in the parent `Sync` element type (not shown in the example).

```
<Replace>
  <CmdID>1234</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Meta><Type xmlns='syncml:metinf'>text/calendar</Type></Meta>
  <Item>
    <Source><LocURI>./20</LocURI></Source>
    <Data>BEGIN:VCALENDAR
VERSION:2.0
METHOD:REQUEST
BEGIN:VEVENT
UID:12345-19991015T133000Z
SEQUENCE:1
DTSTART:19991026T110000Z
DTEND:19991026T190000Z
SUMMARY:Technical Committee Meeting
CATEGORIES:Appointment
ORGANIZER:henry@host.com
ATTENDEES:techcomm@host.com
END:VEVENT
```

```
END:VCALENDAR
    </Data>
  </Item>
</Replace>
```

5.5.12 Results

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

Parent Elements: `SyncBody`

Restrictions: The optional `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 optional `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`.

Exception conditions are not created for this command and there is no requirement to return any request status.

Content Model:

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

Attributes: None

Example: The following is an example of results returned from a `Get` command.

```
<Results>
  <CmdID>4321</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>1234</CmdRef>
  <Meta>
```

```
    <Type xmlns='syncml:metinf'>text/x-vCard</Type>
  </Meta>
  <TargetRef><LocURI>./telecom/pb<LocURI><TargetRef>
  <SourceRef><LocURI>http://www.datasync.com/servlet/</LocURI></SourceRef>
  <Item>
    <Source><LocURI>./1</LocURI></Source>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith, Bruce
TEL;WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
  <Item>
    <Source><LocURI>./2</LocURI></Source>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Ida Blue
N:Blue, Ida
TEL;WORK;VOICE:+1-919-555-2345
END:VCARD
    </Data>
  </Item>
  <Item>
    <Source><LocURI>./3</LocURI></Source>
    <Data>BEGIN:VCARD
VERSION:2.1
FNke McGrath
N:McGrath, Mike
TEL;WORK;VOICE:+1-919-555-3456
END:VCARD
    </Data>
  </Item>
</Results>
```

5.5.13 Search

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

Parent Elements: `SyncBody`

Restrictions: The results from the search are returned in the `Results` command, unless `NoResults` are specified in the command.

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

If specified, the optional `NoResults` element type indicates that the results MUST not be returned for the command. Instead, the results MUST be stored in the temporary storage location specified in the `Target` element type. The temporary results are intended to be specified as the source for a subsequent `Sync` command.

The optional `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

If present, the optional `Target` element type specifies the temporary location on the recipient for the search results. If present, the `NoResults` element MUST also be specified.

One or more `Source` element types MUST be specified. The `Source` element type specifies the databases to be searched.

If present, the optional `Lang` element type specifies the language requested for any search results.

The `Meta` element type MUST be specified. The element type specifies meta-information about the type of search grammar used in the command.

The `Data` element type MUST be specified. The element type specifies the search grammar for the command. The search grammar is generally object-specific and is based on prior agreement or provisioning between the originator and recipient.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the command completed successfully, but there was not any search results, then the (`204`) `No content` exception condition is created by the command.

If the command completed successfully, and the results are being returned in the response and also in subsequent messages, then the (`206`) `Partial content` exception condition is created by the command.

If the command failed because the search grammar was malformed, then the (`400`) `Bad request` exception condition is created by the command.

If the search grammar is not known then (`421`) `Unknown search grammar` MUST be returned.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the originator's authentication credentials specify a principal that has had its rights to issue `Search` commands denied, then the (`403`) `Forbidden` exception condition is created by this command. However, if the recipient does not want to make this fact public, then the (`404`) `Not found` exception condition can be used.

If the recipient does not allow `Search` commands either on the specified database or on the network device, then the (`405`) `Command not allowed` exception condition is created by this command.

If the specified database cannot be found on the recipient network device, then the (`404`) `Not found` exception condition is created by this command.

If the Search command results are too large for processing on the recipient, then the (`413`) `Request entity too large` exception condition is created by this command.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

If there is insufficient space on the recipient database for the temporary results, then the (`420`) `Device full` exception condition is created by the command.

If the media type or format requested for the search results in the `Meta` element type is not supported by the recipient, then the (`415`) `Unsupported media type or format` exception condition is created by the command.

An alternative to the Search command is the use of CGI scripting on `LocURI` within the `Source` and `Target` element types in SyncML commands. See Section 4.18.

Content Model:

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

Attributes: None.

Example:

```
<Search>
  <CmdID>1234</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Source>
    <LocURI>http://www.datasync.org/servlet/syncit/bruce1</LocURI>
  </Source>
  <Meta><Type xmlns='syncml:metinf'>application/sql</Type></Meta>
  <Data>SELECT EQ *
WHERE "FN" EQ "Bruce Smith"
</Data>
</Search>
```

5.5.14 Sequence

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

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

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

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

One or more `Add, Replace, Delete, Copy, Atomic, Map or Sync` element types MUST be specified. These element types MUST be processed in the specified sequence.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the recipient does not support the command, then the (`406`) `Optional feature not supported` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

Content Model:

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

Attributes: None.

Example: The following is an incomplete (i.e., `Add` and `Delete` commands only include skeleton content) example for a `Sequence` command containing two `Add` commands, followed by a `Delete` command.

```
<Sequence>
  <CmdID>1234</CmdID>
  <Add>
    <CmdID>1235</CmdID>
    ...blah, blah...
  </Add>
  <Add>
    <CmdID>1236</CmdID>
    ...blah, blah...
  </Add>
  <Delete>
    <CmdID>1237</CmdID>
    ...blah, blah...
  </Delete>
</Sequence>
```

5.5.15 Sync

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

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

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

If specified, the optional `NoResp` element type indicates that a response status code MUST not be returned for the command.

The `Cred` element type specifies the authentication credential to be used for the command. If not present, the default authentication credential is taken from the `SyncHdr` element type. If a `Cred` element type is not present in any of these other element types, then the command is specified without an authentication credential.

If present, the optional `Target` element type specifies the recipient database to be synchronized.

If present, the optional `Source` element type specifies the originator database to be synchronized.

The optional `Meta` element type specifies meta-information to be used for the command. In this command, the meta-information contains the search grammar. The search grammar is generally object-specific and is based on prior agreement or provisioning between the originator and recipient.

One or more `Add`, `Replace`, `Delete`, `Copy`, `Atomic`, or `Sequence` element types MUST be specified. There is no implied order to the processing of these commands unless they are placed in the `Sequence` command.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the originator's authentication credentials specify a principal that has had its rights to issue `Sync` commands denied, then the (`403`) `Forbidden` exception condition is created by this command. However, if the recipient does not want to make this fact public, then the (`404`) `Not found` exception condition can be used.

If the recipient does not allow `Sync` commands either on the specified database or on the network device, then the (`405`) `Command not allowed` exception condition is created by this command.

If the specified database cannot be found on the recipient network device, then the (404) `Not found` exception condition is created by this command.

If the recipient determines that there is a high probability that the client device data is out of sync, then the (508) `Refresh required` exception condition is created by this command. When this exception condition occurs, the originator of the `Sync` command SHOULD initiate a slow synchronization with the recipient.

Non-specific errors created by the recipient while attempting to complete the command create the (500) `Command failed` exception condition.

Content Model:

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

Attributes: None.

Example: The following is an example of a Sync command with authentication credentials and a single Add of a calendar entry.

```
<Sync>
  <CmdID>1234</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>OGNkNDI1ZTZjNjgwMTNiYWZkOWEyN2JjMjNlZDM4YzENCg==</Data>
  </Cred>
  <Target><LocURI>./mail/bruce1</LocURI></Target>
  <Source><LocURI>./calendar</LocURI></Source>
  <Add>
    <CmdID>1246</CmdID>
    <Item>
      <Source><LocURI>./12</LocURI></Source>
      <Meta>
        <Type xmlns='syncml:metinf'>text/x-vCalendar</Type>
      </Meta>
      <Data>BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
DTSTART:20000531T160000Z
DTEND:20000531T160100Z
SUMMARY:Release v0.9 of specs
END:VEVENT
END:VCALENDAR
      </Data>
    </Item>
  </Add>
</Sync>
```

6 References

[1] vCalendar - The electronic calendaring and scheduling exchange format - Version 1.0, IETF.

[2] Data elements and interchange formats - Information interchange - Representation of dates and times, ISO.

[3] The MD5 Message-Digest Algorithm, IETF.

[4] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, IETF.

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

[6] Tags for the Identification of Languages, IETF.

[7] UTF-8 transformation format of ISO 10646, IETF.

[8] Uniform Resource Identifiers (URI): Generic Syntax, IETF.

[9] SyncML Device Information DTD, SyncML.

[10] SyncML Meta Information DTD, SyncML.

[11] SyncML Synchronization Protocol, SyncML.

[12] vCard - The electronic business card - Version 2.1, IMC.

[13] WAP Binary XML Content Format Specification, WAP Forum.

[14] Extensible Markup Language (XML) 1.0, W3C.

[15] Augmented BNF for Syntax Specifications: ABNF, IETF.

[16] 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), ETSI

7 SyncML DTD

```
<!-- Copyright Notice

Copyright (c) Ericsson, IBM, Lotus, Matsushita Communication Industrial
Co. Ltd., Motorola, Nokia, Palm, Inc., Psion, Starfish Software (2000).
All Rights Reserved.

Implementation of all or part of any Specification may require licenses
under third party intellectual property rights, including without
limitation, patent rights (such a third party may or may not be a
Supporter). The Sponsors of the Specification are not responsible and
shall not be held responsible in any manner for identifying or failing to
identify any or all such third party intellectual property rights.
```

```
THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN ARE PROVIDED ON AN "AS
IS" BASIS WITHOUT WARRANTY OF ANY KIND AND ERICSSON, IBM, LOTUS,
MATSUSHITA COMMUNICATION INDUSTRIAL CO. LTD, MOTOROLA, NOKIA, PALM INC.,
PSION, STARFISH SOFTWARE AND ALL OTHER SYNCML SPONSORS DISCLAIM ALL
WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY
THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY
IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL ERICSSON, IBM, LOTUS, MATSUSHITA COMMUNICATION
INDUSTRIAL CO. LTD, MOTOROLA, NOKIA, PALM INC., PSION, STARFISH SOFTWARE
OR ANY OTHER SYNCML SPONSOR BE LIABLE TO ANY PARTY FOR ANY LOSS OF
PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS,
OR FOR DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR
CONSEQUENTIAL DAMAGES OF ANY KIND IN CONNECTION WITH THIS DOCUMENT OR THE
INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
LOSS OR DAMAGE.

The above notice and this paragraph must be included on all copies of this
document that are made. -->

<!-- This SyncML DTD defines a data synchronization MIME content-type,
called "application/vnd.syncml+xml". The DTD is used by the SyncML
protocol specification. The DTD is to be identified by the URN string
"syncml:syncml". Single element types from this name space can be
referenced as follows:

    <element xmlns='syncml:syncml'>blah, blah</element>

Comments should be sent to syncml@syncml.org -->

<!-- Root or Document Element and -->

<!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)+, 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)>

<!-- 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>

<!-- URI recipient uses 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 specificaiton major/minor version info. -->

<!-- For this version of the DTD, the value is "1.0" -->

<!ELEMENT VerDTD (#PCDATA)>

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

<!-- For example, "xyz/1.0" -->

<!ELEMENT VerProto (#PCDATA)>

<!-- Synchronization data elements -->

<!-- Item element type -->

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

<!-- 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'>chr</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)+)>
```

```
<!-- 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?, 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)+)>

<!-- Status operation. -->

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

<!-- Synchronize Operation. -->

<!ELEMENT Sync (CmdID, NoResp?, Cred?, Target?, Source?, Meta?, (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 [13].

8.1 Code Space Definitions

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

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

The SyncML DTD is assigned the WBXML document public identifier (i.e., the "publicid" WBXML BNF production) associated with the `FD1` 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 1.0//EN
MetInf	01	-//SYNCML//DTD MetInf 1.0//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

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

9 Static Conformance Requirements

Static conformance requirements (SCR) specify the features that are optional, mandatory and recommended within implementations conforming to this specification.

Simple tables are used to specify this information

In these tables, optional features are specified by a "MAY", mandatory features are specified by a "MUST" and recommended features are specified by a "SHOULD".

9.1 Common use elements

The following specifies the static conformance requirements for the SyncML common use elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Archive	MAY	MUST	MAY	MAY
Chal	MUST	MUST	MAY	MUST
Cmd	MUST	MUST	MUST	MUST
CmdID	MUST	MUST	MUST	MUST
CmdRef	MUST	MUST	MUST	MUST
Cred	MUST	MUST	MUST	MUST
Final	MUST	MUST	MUST	MUST
Lang	MAY	MAY	MAY	MAY
LocName	MAY	MAY	MAY	MAY
LocURI	MUST	MUST	MUST	MUST
MsgID	MUST	MUST	MUST	MUST
MsgRef	MUST	MUST	MUST	MUST
NoResp	MUST	MUST	SHOULD	MUST
NoResults	MAY	MAY	MAY	MAY
RespURI	MUST	MAY	MAY	MUST
SessionID*	MUST	MUST	MUST	MUST
SftDel	MUST	MUST	MAY	MUST
Source	MUST	MUST	MUST	MUST
SourceRef	MUST	MUST	MUST	MUST
Target	MUST	MUST	MUST	MUST
TargetRef	MUST	MUST	MUST	MUST
VerDTD	MUST	MUST	MUST	MUST
VerProto	MUST	MUST	MUST	MUST

*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.

9.2 Message container elements

The following specifies the static conformance requirements for the SyncML message container elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
SyncML	MUST	MUST	MUST	MUST
SyncHdr	MUST	MUST	MUST	MUST
SyncBody	MUST	MUST	MUST	MUST

9.3 Data description elements

The following specifies the static conformance requirements for the SyncML data description elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Data	MUST	MUST	MUST	MUST

Item	MUST	MUST	MUST	MUST
Meta	MUST	MUST	MUST	MUST

9.4 Protocol command elements

The following specifies the static conformance requirements for the SyncML protocol command elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Add	MUST	MUST	SHOULD	MUST
Alert	MUST	MUST	MUST	MUST
Atomic	MAY	MAY	MAY	MAY
Copy	MAY	MUST	MAY	MAY
Delete	MUST	MUST	MUST	MUST
Exec	MAY	SHOULD	MAY	MAY
Get*	MUST	MUST	SHOULD	MUST
Map	MAY	MUST	MUST	MAY
MapItem	MAY	MUST	MUST	MAY
Put*	MUST	MUST	MUST	MUST
Replace	MUST	MUST	MUST	MUST
Result*	MUST	MUST	MUST	SHOULD
Search	MAY	MAY	MAY	MAY
Sequence	MAY	MUST	MAY	MAY
Status	MUST	MUST	MUST	MUST
Sync	MUST	MUST	MUST	MUST

*Minimum requirement for a SyncML device is to support Put, Get, and Result when exchanging device information.

10 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

11 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 v1.0 (clear text xml)	http://www.syncml.org/supporters/docs/syncml_devinf_v10_20001207.doc
application/vnd.syncml-devinf+wbxml	SyncML Device Info v1.0 (wbxml)	http://www.syncml.org/supporters/docs/syncml_devinf_v10_20001207.doc
text/plain	Plain Text	http://www.ietf.org/rfc/rfc2046.txt
text/x-vcard	vCard v2.1	Not available
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	Not available
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-relml	SyncML Relational Object	TBD

12 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.

The response status codes defined in this section are also specified in the section that defines the Status command. If there are any differences between these two tables, the definition in this section is the authoritative definition of request status codes.

Implementations that desire to add to these values SHOULD submit a change request to mailto:syncml@syncml.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.
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 also 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	Unauthorized. The requested command failed because proper authentication MUST be provided by the requestor. If no authentication was provided in the request, a suitable challenge MAY also be returned in the Item element type in the Status. 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 doesn't 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	Authentication required. 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 Item 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.
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 ReplyURI 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	Version not supported. The recipient does not support or refuses to support the specified version of SyncML 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 ReplyURI 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 synch with the server.
509	Authentication required. The response code indicates that the server MUST first authenticate with the originator. The client SHOULD also return the suitable challenge in the Item element type in the Status.
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.

13 Alert Types

The alert types in SyncML are a numeric text value. The types are divided into two classes, User Alert, that are intended to be conveyed to the recipient's user agent, and Application Alert, that are intended to be conveyed to a target application on the recipient. 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:syncml@syncml.org.

Alert Code Value	Name	Description
Alert Codes used for user alerts		
100	DISPLAY	Show. The Data element type contains content information that should be processed and displayed through the user agent.
101-150	-	Reserved for future SyncML usage.
Alert Codes used at the synchronization initialization		
200	TWO-WAY	Specifies a client-initiated, two-way sync.
201	SLOW SYNC	Specifies a client-initiated, two-way slow-sync.
202	ONE-WAY FROM CLIENT	Specifies the client-initiated, one-way only sync from the client to the server.
203	REFRESH FROM CLIENT	Specifies the client-initiated, refresh operation for the one-way only sync from the client to the server.
204	ONE-WAY FROM SERVER	Specifies the client-initiated, one-way only sync from the server to the client.
205	REFRESH FROM SERVER	Specifies the client-initiated, refresh operation of the one-way only sync from the server to the client.
Alert Codes used by the server when alerting the sync.		
206	TWO-WAY BY SERVER	Specifies a server-initiated, two-way sync.
207	ONE-WAY FROM CLIENT BY SERVER	Specifies the server-initiated, one-way only sync from the client to the server.

208	REFRESH FROM CLIENT BY SERVER	Specifies the server-initiated, refresh operation for the one-way only sync from the client to the server.
209	ONE-WAY FROM SERVER BY SERVER	Specifies the server-initiated, one-way only sync from the server to the client.
210	REFRESH FROM SERVER BY SERVER	Specifies the server-initiated, refresh operation of the one-way only sync from the server to the client.
211-220	-	Reserved for future SyncML usage.
Special Alert Codes		
221	RESULT ALERT	Specifies a request for sync results.
222	NEXT MESSAGE	Specifies a request for the next message in the package.
223-250	-	Reserved for future SyncML usage.

14 MIME Media Type Registration

The following section is the MIME media type registrations for SyncML specific MIME media types.

14.1 application/vnd.syncml+xml

```
To: ietf-types@iana.org

Subject: Registration of MIME media type application/vnd.syncml+xml

MIME media type name: application

MIME subtype name: vnd.syncml+xml

Required parameters: None

Optional parameters: charset, synctype, verproto, verdtd. May be specified
in any order in the Content-Type MIME header field.

Content-Type MIME header.

charset Parameter

Purpose: Specifies the character set used to represent the SyncML document.
The default character set for SyncML representation protocol is UTF-8, as
defined [RFC 2279].

Formal Specification: The following ABNF defines the syntax for the
parameter.

chrset-param = ";" "charset" "=" <any IANA registered charset identifier>

synctype Parmeter

Purpose: Specifies the data synchronization protocol used by the SyncML
document. If present, the value MUST be the same value as that specified by
```

the "SyncType" element type in the SyncML MIME content information. There is no default value.

Formal Specification: The following ABNF defines the syntax for the parameter.

```
stype-param = ";" "synctype" "=" text
```

verproto Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML synchronization protocol specification for the workflow of messages with SyncML MIME content. If present, MUST be the same value as that specified by the "VerProto" element type in the SyncML MIME content information. If not present, the default value "1.0" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the parameter.

```
verprot-param = ";" "verproto" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"
```

verdtd Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML representation protocol specification that defines the SyncML MIME media type. If present, MUST be the same value as that specified by the "VerDTD" element type in the SyncML MIME content information. If not present, the default value "1.0" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the parameter.

```
verdtd-param = ";" "verdtd" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"
```

Encoding considerations: The default character set for the SyncML MIME content type is UTF-8. Transfer of this character set through some MIME systems may require that the content is first character encoded into a 7bit character set with an IETF character encoding mechanism such as Base64, as defined in RFC2045.

Security considerations:

Authentication: The SyncML MIME content type definition provides for the inclusion of authentication information for the purpose of authenticating the originator and recipient of messages containing the data synchronization content type. The content type definition supports Basic, Base64 userid/password mark-up, MD5 digest challenge and response strings and any other registered authentication credential scheme.

Threats: The SyncML MIME content type definition provides for the inclusion of remote execution commands. Administrators for MIME implementations that

support this content type SHOULD take every standard precaution to assure the activation of the originator of SyncML content, as well as take every standard precaution to confirm the validity of the included remote execution command prior to allowing the command to be executed on the targeted recipient's system.

Interoperability considerations: Implementations that have support for the mandatory features of this content type will greatly increase the chances of interoperating with other implementations supporting this content type. Conformance to this content type requires an implementation to support every mandatory feature.

Published specification: http://www.syncml.org/docs/syncml_v10_20001207.pdf

Applications, which use this media type: This MIME content type is intended for common use by networked data synchronization applications.

Additional information:

Magic number(s): None

File extension(s): XSM

Macintosh File Type Code(s): XSML

Person & email address to contact for further information:
frank.dawson@nokia.com

Intended usage: COMMON

Author/Change controller: frank.dawson@nokia.com

14.2 application/vnd.syncml+wbxml

To: ietf-types@iana.org

Subject: Registration of MIME media type application/vnd.syncml+wbxml

MIME media type name: application

MIME subtype name: vnd.syncml+wbxml

Required parameters: None

Optional parameters: charset, synctype, verproto, verdtd. May be specified in any order in the Content-Type MIME header field.

Content-Type MIME header.

charset Parameter

Purpose: Specifies the character set used to represent the SyncML document. The default character set for SyncML representation protocol is UTF-8, as defined [RFC 2279].

Formal Specification: The following ABNF defines the syntax for the parameter.

```
chrset-param = ";" "charset" "=" <any IANA registered charset identifier>

synctype Parmeter

Purpose: Specifies the data synchronization protocol used by the SyncML
document. If present, the value MUST be the same value as that specified by
the "SyncType" element type in the SyncML MIME content information. There is
no default value.

Formal Specification: The following ABNF defines the syntax for the
parameter.

stype-param = ";" "synctype" "=" text

verproto Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML
synchronization protocol specification for the workflow of messages with
SyncML MIME content. If present, MUST be the same value as that specified by
the "VerProto" element type in the SyncML MIME content information. If not
present, the default value "1.0" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the
parameter.

verprot-param = ";" "verproto" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"

verdtd Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML
representation protocol specification that defines the SyncML MIME media
type. If present, MUST be the same value as that specified by the "VerDTD"
element type in the SyncML MIME content information. If not present, the
default value "1.0" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the
parameter.

verdtd-param = ";" "verdtd" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"

Encoding considerations: The default character set for the SyncML MIME
content type is UTF-8. Transfer of this character set through some MIME
systems may require that the content is first character encoded into a 7bit
character set with an IETF character encoding mechanism such as Base64, as
defined in RFC2045.

Security considerations:

Authentication: The SyncML MIME content type definition provides for the
inclusion of authentication information for the purpose of authenticating
the originator and recipient of messages containing the data synchronization
```

content type. The content type definition supports Basic, Base64 userid/password mark-up, MD5 digest challenge and response strings and any other registered authentication credential scheme.

Threats: The SyncML MIME content type definition provides for the inclusion of remote execution commands. Administrators for MIME implementations that support this content type SHOULD take every standard precaution to assure the authentication of the originator of SyncML content, as well as take every standard precaution to confirm the validity of the included remote execution command prior to allowing the command to be executed on the targeted recipient's system.

Interoperability considerations: Implementations that have support for the mandatory features of this content type will greatly increase the chances of interoperating with other implementations supporting this content type. Conformance to this content type requires an implementation to support every mandatory feature.

Published specification: http://www.syncml.org/docs/syncml_v10_20001207.pdf

Applications, which use this media type: This MIME content type is intended for common use by networked data synchronization applications.

Additional information:

Magic number(s): None

File extension(s): BSM

Macintosh File Type Code(s): BSML

Person & email address to contact for further information:
frank_dawson@nokia.com

Intended usage: COMMON

Author/Change controller: frank.dawson@nokia.com