SyncML Representation Protocol,

Data Synchronization Usage, version 1.1.2

Approved Version 12-June-2003

Open Mobile Alliance
OMA-SyncML-DataSyncRep-V1_1_2-20030612-A

Continues the Technical Activities
Originated in the SyncML Initiative

Contents

1. SCOPE 5
2. REFERENCES 6
   2.1 NORMATIVE REFERENCES 6
   2.2 INFORMATIVE REFERENCES 7
3. TERMINOLOGY AND CONVENTIONS 8
   3.1 CONVENTIONS 8
   3.2 DEFINITIONS 8
   3.3 ABBREVIATIONS 9
4. INTRODUCTION 10
5. SYNCML DATA SYNCHRONIZATION USAGE 11
   5.1 SYNCML DATA SYNCHRONIZATION FRAMEWORK 11
   5.2 SYNCML DATA FORMATS 12
   5.3 CAPABILITIES EXCHANGE 12
   5.4 DATA IDENTIFIER MAPPING 13
   5.5 REFRESHING DATA 13
   5.6 SOFT AND HARD DATA DELETION 13
   5.7 ARCHIVING DATA 13
   5.8 REPLACING DATA 13
   5.9 SEARCHING FOR DATA 14
   5.10 LOCALIZATION 14
   5.11 MIME USAGE 14
   5.12 TARGET AND SOURCE ADDRESSING 14
   5.13 TARGET ADDRESS FILTERING 17
      5.13.1 Arbitrary Database Object Filter 19
      5.13.2 XML Document Filter 19
      5.13.3 Contacts Media Object Filter 20
      5.13.4 Calendar Media Object Filter 20
      5.13.5 Email Media Object Filter 21
6. MARK-UP LANGUAGE DESCRIPTION 22
   6.1 COMMON USE ELEMENTS 22
      6.1.1 Archive 22
      6.1.2 Chal 22
      6.1.3 Cmd 23
      6.1.4 CmdID 23
      6.1.5 CmdRef 24
      6.1.6 Cred 24
      6.1.7 Final 24
      6.1.8 Lang 25
      6.1.9 LocName 25
      6.1.10 LocURI 25
      6.1.11 MoreData 25
      6.1.12 MsgID 26
      6.1.13 MsgRef 26
      6.1.14 NoResp 27
      6.1.15 NoResults 27
      6.1.16 NumberOfChanges 28
      6.1.17 RespURI 28
      6.1.18 SessionID 29
      6.1.19 SftDel 29
      6.1.20 Source 30
      6.1.21 SourceRef 31
      6.1.22 Target 31

6.1.23 TargetRef 32
6.1.24 VerDTD 33
6.1.25 VerProto 33
6.2 MESSAGE CONTAINER ELEMENTS 33
6.2.1 SyncML 33
6.2.2 SyncHdr 34
6.2.3 SyncBody 34
6.3 DATA DESCRIPTION ELEMENTS 35
6.3.1 Data 35
6.3.2 Item 36
6.3.3 Meta 36
6.4 PROTOCOL MANAGEMENT ELEMENTS 37
6.4.1 Status 37
6.5 PROTOCOL COMMAND ELEMENTS 37
6.5.1 Add 37
6.5.2 Alert 39
6.5.3 Atomic 40
6.5.4 Copy 41
6.5.5 Delete 43
6.5.6 Exec 44
6.5.7 Get 46
6.5.8 Map 48
6.5.9 MapItem 49
6.5.10 Put 49
6.5.11 Replace 51
6.5.12 Results 52
6.5.13 Search 53
6.5.14 Sequence 55
6.5.15 Sync 56
APPENDIX A. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 65
APPENDIX B. CHANGE HISTORY (INFORMATIVE) 65

1. Scope

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

SyncML Components

SyncML is a specification that contains the following main components:

- An XML-based representation protocol
- A synchronization protocol and a device management protocol
- Transport bindings for the protocol

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

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

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

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

SyncML is both data type and data store independent. SyncML can carry any data type which can be represented as a MIME object. To promote interoperability between different implementations of SyncML, the specification includes the representation formats used for common PIM data. The SyncML Representation Protocol [REPPRO] specifies the common XML syntax and semantics used by all SyncML protocols. The SyncML representation protocol is defined by a set of messages that are conveyed between entities participating in a SyncML operation. This document gives examples and explains the restrictions on using the common representation protocol for synchronizing data. The SyncML Representation Protocol, Device Management Usage [DMREPU] provides information on using the common representation protocol for device management.

2. References

2.1 Normative References

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

[IMCVCAL] "vCalendar – The electronic calendaring and scheduling exchange format – Version 1.0", URL:http://www.imc.org/pdi/vcal-10.doc

[IMCVCARD] "vCard - The electronic business card - Version 2.1", URL:http://www.imc.org/pdi/vcard-21.doc

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

[SYNCMETA] "SyncML Meta Information," Open Mobile Alliance™, OMA-SyncML-Meta-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[REPPRO] "SyncML Representation Protocol", Open Mobile Alliance™, OMA-SyncML-RepPro-V1_1_2, URL:http://www.openmobilealliance.org/tech/docs

[RFC822] "Standard for the format of ARPA Internet text messages", David H. Crocker, August 1982, URL:http://www.ietf.org/rfc/rfc822.txt

[RFC1766] "Tags for the Identification of Languages", H. Alvestrand, March 1995, URL:http://www.ietf.org/rfc/rfc1766.txt

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

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

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF", D. Crocker, Ed., P. Overell, November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt

[RFC2279] "UTF-8, a transformation format of ISO 10646", F. Yergeau, January 1998, URL:http://www.ietf.org/rfc/rfc2279.txt

[RFC2445] "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", F. Dawson, D. Stenerson, November 1998, URL:http://www.ietf.org/rfc/rfc2445.txt

[SYNCDEVDTD] "SyncML Device Information DTD", Open Mobile Alliance™, OMA-SyncML-DevInfo-V1_1_2", URL:http://www.openmobilealliance.org/syncml/technology.html

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

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

2.2 Informative References

None.

3. Terminology and Conventions

3.1 Conventions

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

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

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

3.2 Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Originator - The network device that creates a SyncML request.

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

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

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

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

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

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

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

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

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

3.3 Abbreviations

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

4. Introduction

SyncML Data Synchronization is a specification for a common data synchronization framework and XML-based format, or representation protocol, for synchronizing data on networked devices. SyncML Data Synchronization 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 Data Synchronization can also be used for peer-to-peer data synchronization. SyncML Data Synchronization 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.

5. SyncML Data Synchronization Usage

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.

5.1 SyncML Data Synchronization Framework

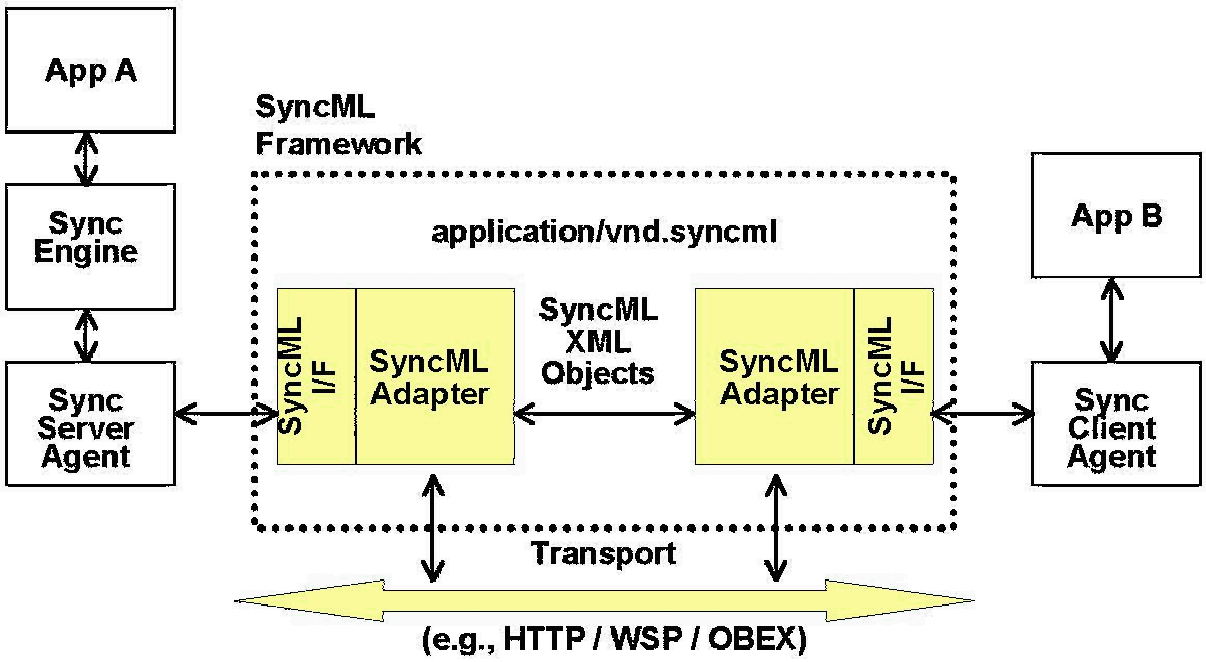
SyncML Data Synchronization not only defines a format, but also a conceptual data synchronization framework and data synchronization protocol. The framework is depicted in Figure 1. In the figure, the scope of the SyncML Data Synchronization Framework is shown by the dotted-line box. The Framework consists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

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

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



5.2 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.

5.3 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 [SYNCDEV] for further details on the specification of the Device Information DTD.

5.4 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.

5.5 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.

5.6 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.

5.7 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 `Archive` would generate an error condition (i.e., `(210) Delete` without Archive.).

5.8 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 [SYNCMETA] 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.

5.9 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.

5.10 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 [RFC2279].

5.11 MIME Usage

There are two MIME content types for the SyncML Data Synchronization 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 8 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 Data Synchronization Messages within transport and session level protocols that support MIME content types.

5.12 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. 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 MAY 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 MAY 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 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 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`.

5.13 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 imposes a temporary constraint on the set of data items returned by its parent command. The filter itself has no effect on the set of data items, which is synchronized (i.e. previously synchronized items which are not included in the filter result set remain in the set of synchronized data). 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. When the package containing a Sync command using target address filtering for a given database requires multiple messages, the CGI parameter MUST be included every time the Sync command for that database appears in a message in order to maintain the filtering restriction.

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 [RFC2234]].

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
```

5.13.1 Arbitrary Database Object Filter

The filter for accessing individual items from an arbitrary 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>
```

5.13.2 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>
```

5.13.3 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
[IMCVCARD] 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>
```

5.13.4 Calendar Media Object Filter

The filter for calendar media objects makes use of the iCalendar RFC2445 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 RFC2445
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 [ISO8601]]. The values of "RRULE" are a recurrence rule as specified in [[IMCVCAL]. 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 [IMCVCAL]. The value of "UID", "URL", "RECURID" are any valid value as specified for these properties in [IMCVCAL]. 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>
```

5.13.5 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>
```

6. Mark-up Language Description

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.

Restrictions listed in this document are in addition to the restrictions listed in [REPPRO].

6.1 Common Use Elements

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

6.1.1 Archive

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 210 (`Delete` without `Archive`) MUST be returned if the delete was successful. See `Delete` for other possible error codes.

Example:

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

6.1.2 Chal

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.1.3 Cmd

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.1.4 CmdID

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<Add>
  <CmdID>1234</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.1.5 CmdRef

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.1.6 Cred

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example: The following is an example of a MD5 digest authentication credential scheme where the user name is "Bruce2", the password is "OhBehave", and the nonce is "Nonce". Let MD5(data) denote the result of applying the MD5 hash algorithm to "data". In this example, the MD5 Digest is MD5(MD5("Bruce2: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>Zz6EivR3yeaaENcRN6lpAQ==</Data>
</Cred>
```

6.1.7 Final

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>...blah, blah...</SyncHdr>
  <SyncBody>
    ...blah, blah...
    <Final/>
  </SyncBody>
</SyncML>
```

6.1.8 Lang

Restrictions: The value of the element type MUST be a valid [RFC1766] formatted International language identifier. The semantics for this element type are quite different than the `xml:lang` attribute defined in [XML], 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.

Example:

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

6.1.9 LocName

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

6.1.10 LocURI

Restrictions: Section 4.12, "Target and Source Addressing" provides restrictions on the values for the `LocURI` element type.

Example:

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>SyncML/1.1</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.datasync.org/servlet/syncit/</LocURI></Target
>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>
```

6.1.11 MoreData

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<Add>
  <CmdID>15</CmdID>
  <Meta>
    <Type>text/x-vcard</Type>
    <size>3000</size>
  </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:+1-919-555-9876
NOTE: here starts a huge note field, or icon etc...
    </Data>
    <MoreData/>
  </Item>
</Add>
```

6.1.12 MsgID

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>1.1</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.syncml.host.com/</LocURI></Target>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>
```

6.1.13 MsgRef

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

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

6.1.14 NoResp

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.

Example:

```
<Replace>
  <CmdID>1</CmdID>
  <NoResp/>
  <Item>
    <Source><LocURI>./127</LocURI></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
ADR:;;123 Main St.;Anywhere;CA;;US
END:VCARD
    </Data>
  </Item>
</Replace>
```

6.1.15 NoResults

Restrictions: The indicator is used to force the results of a 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 condition will be created.

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>
```

6.1.16 NumberOfChanges

Restrictions: The element type SHOULD be specified by the server, but only if the client has indicated that it supports NumberOfChanges. It MAY be specified by the client. If synchronizations are carried out on more than one datastore (e.g. Contacts & Calendar), then <NumberOfChanges> should be specified for each datastore.

The <NumberOfChanges> element MUST only be specified in the <Sync> command.

Example:

```
<Sync>

  <CmdID>5</CmdID>

  <Target><LocURI>contacts</LocURI></Target>

  <Source><LocURI>C:\System\data\Contacts.cdb</LocURI></Source>

  <NumberOfChanges>20</NumberOfChanges>

  ...

</Sync>
```

6.1.17 RespURI

Restrictions: 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) [ISO8601]. Additional script parameter can be prefixed or appended to this SyncML specific CGI script parameter. See Section 4.13.

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>1.1</VerDTD>
  <VerProto>SyncML/1.1</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>
```

6.1.18 SessionID

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>
    <VerDTD>1.1</VerDTD>
    <VerProto>SyncML/1.1</VerProto>
    <SessionID>1</SessionID>
    <MsgID>3</MsgID>
    <Target>
      <LocURI>IMEI:001004FF1234567</LocURI>
    </Target>
    <Source>
      <LocURI>http://www.datasync.org/servlet/syncit/</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
  ...blah, blah...
  </SyncBody>
</SyncML>
```

6.1.19 SftDel

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 so 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.

Example:

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

6.1.20 Source

Restrictions: Section 4.11, "Target and Source Addressing" provides semantics on the content of the `Source` element type.

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.

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

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>SyncML/1.1</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>
```

6.1.21 SourceRef

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

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

6.1.22 Target

Restrictions: Section 4.11, "Target and Source Addressing" provides semantics on the content of the `Target` element type.

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.

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

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>SyncML/1.1</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>
```

6.1.23 TargetRef

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

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

6.1.24 VerDTD

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>SyncML/1.1</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>
```

6.1.25 VerProto

Restrictions: Major revisions 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 1.1.

Example:

```
<SyncHdr>
  <VerDTD>1.1</VerDTD>
  <VerProto>SyncML/1.1</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>
```

6.2 Message Container Elements

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

6.2.1 SyncML

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).

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>
    <VerDTD>1.1</VerDTD>
    <VerProto>SyncML/1.1</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.datasync.org/servlet/syncit</LocURI>

    </Target>
    <Source>

      <LocURI>IMEI:001004FF1234567</LocURI>

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

6.2.2 SyncHdr

Restrictions: 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.

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>
    <VerDTD>1.1</Version>
    <VerProto>SyncML/1.1</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>
```

6.2.3 SyncBody

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.1'>
  <SyncHdr>
    <VerDTD>1.1</VerDTD>
    <VerProto>SyncML/1.1</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>Zz6EivR3yeaaENcRN6lpAQ==</Data>
    </Cred>
  </SyncHdr>
  <SyncBody>
    <Get>
      <CmdID>1234</CmdID>
      <Item>
        <Target><LocURI>./devinf10</LocURI></Target>
      </Item>
    </Get>
  </SyncBody>
</SyncML>
```

6.3 Data Description Elements

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

6.3.1 Data

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.3.2 Item

Restrictions: No additional restrictions beyond those defined in [REPPRO].

Example:

```
<Add>
  <CmdID>1</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.3.3 Meta

Restrictions:

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 Meta information, e.g. the type of search grammar.

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>
```

6.4 Protocol Management Elements

6.4.1 Status

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.5 Protocol Command Elements

6.5.1 Add

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

Example:

```
<Add>
  <CmdID>12345</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.2 Alert

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 mandatory `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 MUST 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 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.3 Atomic

Restrictions: The mandatory `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.

Nested Atomic commands are not legal. A nested Atomic command will generate an error (500) - `command failed.`

If a client can execute all the atomic commands together (and thus guarantee the result) then a client MAY split the responses up over multiple messages. If a client cannot execute all the atomic commands together (and thus cannot guarantee the results of commands not executed) and status responses would go into multiple messages, then the Atomic command MUST fail with status code `(517)Atomic response too large to fit in message.` Previously executed commands in Atomic command MUST be rolled back.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>Zz6EivR3yeaaENcRN6lpAQ==</Data>
    </Cred>
    <Item>
      <Target><LocURI>./devinf10/version</LocURI></Target>
      <Data>20000401T133000Z</Data>
    </Item>

  </Replace>
</Atomic>
```

6.5.4 Copy

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 mandatory `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 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 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 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.

Example:

```
<Copy>
  <CmdID>12345</CmdID>
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.5 Delete

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 mandatory `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 and supported, the optional `Archive` element type indicates that the recipient MUST 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 `(211) Item not deleted` 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Item>
    <Target><LocURI>./5</LocURI></Target>
  </Item>
  <Item>
    <Target><LocURI>./10</LocURI></Target>
  </Item>
  <Item>
    <Target><LocURI>./8</LocURI></Target>
  </Item>
</Delete>
```

6.5.6 Exec

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 implementers 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 mandatory `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 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.7 Get

Restrictions: There is no synchronization state information for data retrieved using the `Get` command. This command MUST NOT be specified within a `Sync` command.

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

The mandatory `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 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.8 Map

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 idempotent, 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 mandatory `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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.9 MapItem

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.

Example:

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

6.5.10 Put

Restrictions: There is no synchronization state information for data transferred using the `Put` command. This command MUST NOT be specified within a `Sync` command.

The mandatory `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 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
  </Item>

</Put>
```

6.5.11 Replace

Restrictions: The `Replace` command is used to replace data on the recipient. This command MUST only be specified within a `Sync` command.

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 mandatory `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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.12 Results

Restrictions: No additional restrictions beyond those defined in [REPPRO].

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>
```

6.5.13 Search

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

The mandatory `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 4.13.

Example:

```
<Search>
  <CmdID>1234</CmdID>
```

```
  <Cred>
    <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
    </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

6.5.14 Sequence

Restrictions: The mandatory `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.

Nested Sequence commands are not legal. A nested Sequence command will generate an error `(500)Command failed.`

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>
```

6.5.15 Sync

Restrictions: The mandatory `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.

Zero 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.

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>Zz6EivR3yeaaENcRN6lpAQ==</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>
```

7. 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:technical-comments@openmobilealliance.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 synchronization.
201	SLOW SYNC	Specifies a client-initiated, two-way slow-synchronization.
202	ONE-WAY FROM CLIENT	Specifies the client-initiated, one-way only synchronization from the client to the server.
203	REFRESH FROM CLIENT	Specifies the client-initiated, refresh operation for the one-way only synchronization from the client to the server.
204	ONE-WAY FROM SERVER	Specifies the client-initiated, one-way only synchronization from the server to the client.
205	REFRESH FROM SERVER	Specifies the client-initiated, refresh operation of the one-way only synchronization from the server to the client.
Alert Codes used by the server when alerting the synchronization		
206	TWO-WAY BY SERVER	Specifies a server-initiated, two-way synchronization.
207	ONE-WAY FROM CLIENT BY SERVER	Specifies the server-initiated, one-way only synchronization from the client to the server.
208	REFRESH FROM CLIENT BY SERVER	Specifies the server-initiated, refresh operation for the one-way only synchronization from the client to the server.
209	ONE-WAY FROM SERVER BY SERVER	Specifies the server-initiated, one-way only synchronization from the server to the client.
210	REFRESH FROM SERVER BY SERVER	Specifies the server-initiated, refresh operation of the one-way only synchronization from the server to the client.
211-220	-	Reserved for future SyncML usage.
Special Alert Codes		

221	RESULT ALERT	Specifies a request for synchronization results.
222	NEXT MESSAGE	Specifies a request for the next message in the package.
223	NO END OF DATA	End of Data for chunked object not received
224-250	-	Reserved for future SyncML usage.

8. MIME Media Type Registration

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

8.1 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 in [RFC2279].

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

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

synctype Parameter

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 "1.1" 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 "1.1" 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: URL:http://www.openmobilealliance.org/tech/docs

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:
admins@syncml.org

Intended usage: COMMON

Author/Change controller: mailto:technical-comments@openmobilealliance.org

8.2 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 [RFC2279]].

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

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

synctype Parameter

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 "1.1" 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 "1.1" 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_sync_represent_v111_20021002.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:
admins@syncml.org

Intended usage: COMMON

Author/Change controller: mailto:technical-comments@openmobilealliance.org

Appendix A. Static Conformance Requirements (Normative)

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", as defined in [RFC2119].

An implementation that does not include a particular option MUST be prepared to interoperate with another implementation that does include the option though perhaps with reduced functionality.

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	MAY	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
MoreData	MUST	MUST	MAY	MAY
MsgID	MUST	MUST	MUST	MUST
MsgRef	MUST	MUST	MUST	MUST
NoResp	MAY	MUST	MAY	MUST
NoResults	MAY	MAY	MAY	MAY
NumberOfChanges	MAY	MUST	MAY	MAY

RespURI	MAY	MUST	MAY	MUST
SessionID*	MUST	MUST	MUST	MUST
SftDel	MAY	MAY	MAY	MAY
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.

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

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

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.

Content formats

NOTE: If a server supports a data type listed below, it must also support the associated content format.

Data Type	Content Format	Status
Contact	vCard 2.1	MUST
	vCard 3.0	SHOULD
Calendar	vCalendar 1.0	MUST

	iCalendar 2.0	SHOULD
Memos	text/plain	MUST
Tasks	vTodo 1.0	MUST
Email	message/rfc822	MUST
	message/rfc2822	MUST
	Message/rfc2045	MUST

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
OMA-SyncML-DataSyncRep-V1_1_2-20030612-A	12 June 2003	Approved by TP. TP ref# OMA-TP-2003-0264R1