



SyncML Representation Protocol, Data Synchronization Usage

Candidate Version 1.2 – 06 Jun 2004

Open Mobile Alliance
OMA-SyncML-DataSyncRep-V1_2-20040601-C

Use of this document is subject to all of the terms and conditions of the Use Agreement located at <http://www.openmobilealliance.org/UseAgreement.html>.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the “OMA IPR Declarations” list at <http://www.openmobilealliance.org/ipr.html>. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR'S REPRESENTED ON THE “OMA IPR DECLARATIONS” LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

© 2004 Open Mobile Alliance Ltd. All Rights Reserved.

Used with the permission of the Open Mobile Alliance Ltd. under the terms set forth above.

Contents

1. Scope	6
2. References.....	8
2.1 Normative References.....	8
2.2 Informative References	9
3. Terminology and Conventions.....	10
3.1 Conventions	10
3.2 Definitions.....	10
3.3 Abbreviations	11
4. Introduction	12
5. SyncML Data Synchronization Usage	13
5.1 SyncML Data Synchronization Framework.....	13
5.2 SyncML Data Formats	14
5.3 Capabilities Exchange	14
5.4 Data Identifier Mapping.....	15
5.5 Refreshing Data.....	15
5.6 Soft and Hard Data Deletion.....	15
5.7 Archiving Data	15
5.8 Replacing Data	15
5.8.1 Field-level Replace	15
5.9 Searching For Data.....	16
5.10 Localization	16
5.11 MIME Usage	16
5.12 Target and Source Addressing.....	16
5.13 Data Sync Record and Field Level Filtering.....	16
5.13.1 Filter Behavior Definition.....	16
5.13.2 Filter Query Syntax.....	16
5.13.3 Indicating Filter Support.....	16
5.13.4 Handling Data Outside Filter Criteria.....	16
5.13.5 Examples.....	16
6. Mark-up Language Description	16
6.1 Common Use Elements.....	16
6.1.1 Archive	16
6.1.2 Chal.....	16
6.1.3 Cmd.....	16
6.1.4 CmdID	16
6.1.5 CmdRef.....	16
6.1.6 Cred.....	16
6.1.7 Field.....	16
6.1.8 Filter.....	16

6.1.9	FilterType.....	16
6.1.10	Final.....	16
6.1.11	Lang.....	16
6.1.12	LocName.....	16
6.1.13	LocURI.....	16
6.1.14	MoreData.....	16
6.1.15	MsgID.....	16
6.1.16	MsgRef.....	16
6.1.17	NoResp.....	16
6.1.18	NoResults.....	16
6.1.19	NumberOfChanges.....	16
6.1.20	Record.....	16
6.1.21	RespURI.....	16
6.1.22	SessionID.....	16
6.1.23	SftDel.....	16
6.1.24	Source.....	16
6.1.25	SourceParent.....	16
6.1.26	SourceRef.....	16
6.1.27	Target.....	16
6.1.28	TargetParent.....	16
6.1.29	TargetRef.....	16
6.1.30	VerDTD.....	16
6.1.31	VerProto.....	16
6.2	Message Container Elements.....	16
6.2.1	SyncML.....	16
6.2.2	SyncHdr.....	16
6.2.3	SyncBody.....	16
6.3	Data Description Elements.....	16
6.3.1	Data.....	16
6.3.2	Item.....	16
6.3.3	Meta.....	16
6.4	Protocol Management Elements.....	16
6.4.1	Status.....	16
6.5	Protocol Command Elements.....	16
6.5.1	Add.....	16
6.5.2	Alert.....	16
6.5.3	Atomic.....	16
6.5.4	Copy.....	16
6.5.5	Delete.....	16
6.5.6	Exec.....	16
6.5.7	Get.....	16
6.5.8	Map.....	16
6.5.9	MapItem.....	16
6.5.10	Move.....	16
6.5.11	Put.....	16
6.5.12	Replace.....	16
6.5.13	Results.....	16
6.5.14	Search.....	16
6.5.15	Sequence.....	16
6.5.16	Sync.....	16
7.	<i>Alert Types.....</i>	<i>16</i>
8.	<i>Base Media and Content formats.....</i>	<i>16</i>
9.	<i>MIME Media Type Registration.....</i>	<i>16</i>

9.1	application/vnd.syncml+xml	16
9.2	application/vnd.syncml+wbxml	16
<i>Appendix A.</i>	<i>Static Conformance Requirements (Normative)</i>	<i>16</i>
A.1	Client Data Sync Usage of SyncML Representation	16
A.2	Server Data Sync Usage of SyncML Representation	16
<i>Appendix B.</i>	<i>Change History (Informative)</i>	<i>16</i>
B.1	Approved Version History	16
B.2	Draft/Candidate Version version 1.2 History	16

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.

The SyncML Initiative merged with the Open Mobile Alliance in November 2002. The SyncML legacy specifications were converted to the OMA format with the 1.1.2 versions of OMA SyncML Common, OMA Data Synchronization and OMA Device Management in May 2002. The relationship between these documents which had been created during the SyncML Initiative has been preserved and is depicted in Figure 1 OMA SyncML Specification Structure and Relationships.

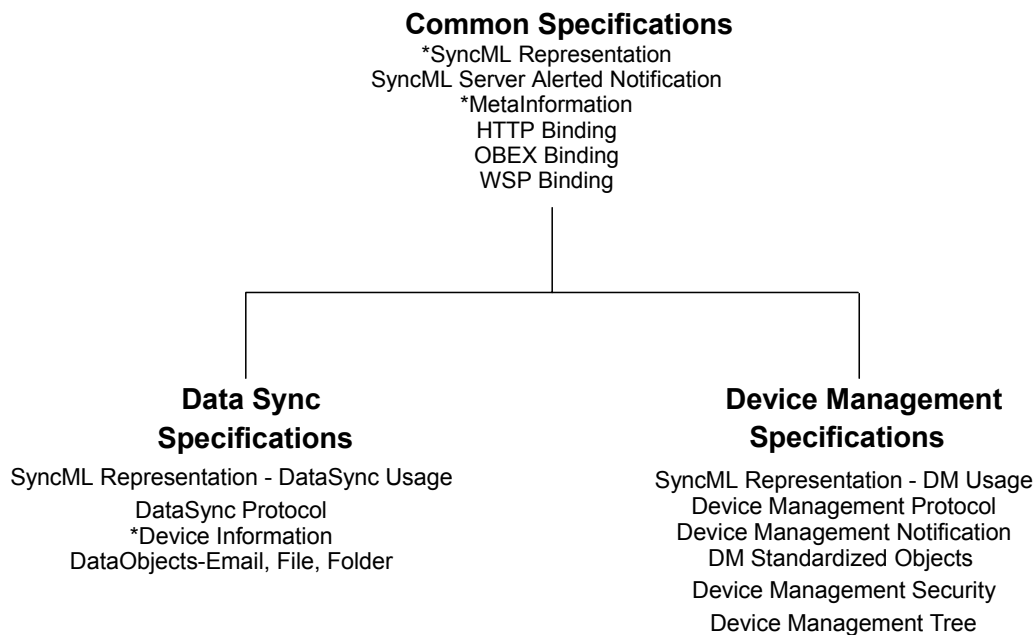


Figure 1 OMA SyncML Specification Structure and Relationships

The OMA SyncML Common Specifications Enabler Release includes the following documents:

- SyncML Representation: The XML-based representation protocol which specifies the common XML syntax and semantics used by all SyncML protocols. This defines the superset of the DS and DM representation protocols. (* includes DTD).
- The transport bindings: HTTP, OBEX, WSP. These specify the features REQUIRED for each transport to send and receive DS and DM protocol messages.
- The Meta Information associated with a SyncML command or data item or collection used by either DS or DM (* includes DTD)
- SyncML Server Alerted Notification: The logical structure and format of the notification messages used by all SyncML server alerted notifications, for both DS and DM.

The OMA Data Synchronization Specifications Enabler Release includes the following documents:

- SyncML Representation DataSync Usage: The subset of the Common Specifications SyncML Representation Specification necessary to define the Data Synchronization commands and protocol, with examples and commentary specific to DS.
- DataSyncProtocol: Specifies how SyncML Common messages conforming to the DTD are exchanged in order to allow an OMA DS client and server to exchange additions, deletions, updates and other status information.
- Device Information: Used to exchange device specific information, including hardware, firmware, software levels, available memory, and local databases supported. (* Includes DTD)
- Data Objects: Email, File, Folder: Each object is identified by a unique MIME media type (eg. **application/vnd.omads-email**). The objects are either represented by or encapsulated in a mark-up language defined by xml. Meta or state data is included in the representation (eg. Read/Unread, Creation Date, Last Modified Date).

Although the SyncML Common specification defines transport bindings that specify how to use a particular transport to exchange messages and responses, the SyncML Common representation, synchronization and device management protocols are transport-independent. Each package in these protocols 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 Common could not be implemented using email or message queues, to list only two alternatives. Because the SyncML Common messages are self-contained, multiple transports could 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 Common based on the WAP Forum's WBXML is defined. Messages may also be passed in clear text if desired. In this and other ways SyncML Common addresses the bandwidth and resource limitations imposed by mobile devices.

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

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](http://www.openmobilealliance.org)
- [IMCVCAL] “vCalendar – The electronic calendaring and scheduling exchange format – Version 1.0”, [URL:http://www.imc.org/pdi/vcal-10.doc](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](http://www.imc.org/pdi/vcard-21.doc)
- [IOPPROC] “OMA Interoperability Policy and Process”, Open Mobile Alliance™, OMA-IOP-Process-V1_2_0, [URL:http://www.openmobilealliance.org](http://www.openmobilealliance.org)
- [ISO8601] “Data elements and interchange formats – Information interchange – Representation of dates and times ISO 8601-2000”, [URL://www.iso.ch/iso/en/ISOOnline.openerspage](http://www.iso.ch/iso/en/ISOOnline.openerspage)
- [REPPRO] “SyncML Representation Protocol”, Open Mobile Alliance™, OMA-SyncML-RepPro-V1_2, [URL:http://www.openmobilealliance.org](http://www.openmobilealliance.org)
- [RFC822] “Standard for the format of ARPA Internet text messages”, David H. Crocker, August 1982, [URL:http://www.ietf.org/rfc/rfc822.txt](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](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](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](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](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](http://www.ietf.org/rfc/rfc2279.txt)
- [RFC2426] “vCard MIME Directory Profile”, F. Dawson, T. Howes, September, 1998, [URL:http://www.ietf.org/rfc/rfc2426.txt](http://www.ietf.org/rfc/rfc2426.txt)
- [RFC2445] “Internet Calendaring and Scheduling Core Object Specification (iCalendar)”, F. Dawson, D. Stenerson, November 1998, [URL:http://www.ietf.org/rfc/rfc2445.txt](http://www.ietf.org/rfc/rfc2445.txt)
- [SYNCDEVDTD] “SyncML Device Information DTD”, Open Mobile Alliance™, OMA-SyncML-DevInfo-V1_2”, [URL:http://www.openmobilealliance.org](http://www.openmobilealliance.org)

[URL:http://www.openmobilealliance.org/syncml/technology.html](http://www.openmobilealliance.org/syncml/technology.html)

- [SYNCMETA] “SyncML Meta Information,” Open Mobile Alliance™, OMA-SyncML-Meta-V1_2,
[URL:http://www.openmobilealliance.org](http://www.openmobilealliance.org)
- [SYNCPRO] “SyncML Synchronization Protocol”, Open Mobile Alliance™, OMA-SyncML-DataSyncProtocol-V1_2”, [URL:http://www.openmobilealliance.org](http://www.openmobilealliance.org)
- [XML] “Extensible Markup Language (XML) 1.0”, World Wide Web Consortium Recommendation,
[URL:http://www.w3.org/TR/REC-xml](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 {{i1, data1}, ... {in, datan}}}). In SyncML, data collections are synchronized with each other. See data element.
data element	A piece of data and an associated identifier for the data, (e.g., {i, data}).
Data element equivalence	When two data elements are synchronized. The exact semantics is defined by a given data synchronization model.
Data exchange	The act of sending, requesting or receiving a set of data elements.
Data format	The encoding used to format a data type. For example, characters or integers or character encoded binary data.
Data type	The schema used to represent a data object (e.g., text/calendar MIME content type for an iCalendar representation of calendar information or text/directory MIME content type for a vCard representation of contact information).
Data synchronization	The act of establishing an equivalence between two data collections, where each data element in one item maps to a data item in the other, and their data is equivalent.
Data synchronization protocol	The well-defined specification of the "handshaking" or workflow 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 necessary to support SyncML. However, a SyncML implementation that integrates an XML parser might be easier to enhance. This document assumes that the reader has some familiarity with XML syntax and terminology.
Recipient	The network device that receives a SyncML request, processes the request and sends any resultant SyncML response.
Representation protocol	A well-defined format for exchanging a particular form of information. SyncML is a representation protocol for conveying data synchronization and device management operations.
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.
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.
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.

3.3 Abbreviations

DTD	Document Type Definition
GUID	Global Unique Identifier
HTTP	HyperText Transfer Protocol
IMEI	International Mobile Equipment Identifier
LUID	Local Unique Identifier
MSC	Message Sequence Chart
MSG	Message
OBEX	OBject Exchange protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WSP	Wireless Session Protocol

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

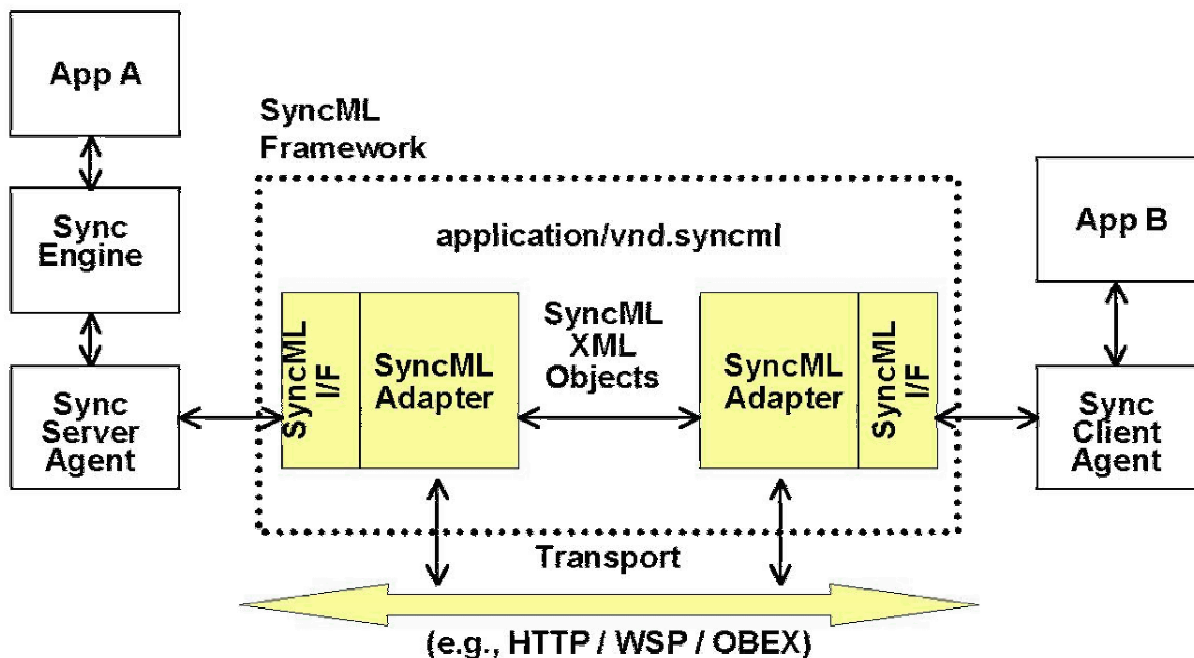
SyncML Data Synchronization not only defines a format, but also a conceptual data synchronization framework and data synchronization protocol. The framework is depicted in Figure 1. In the figure, the scope of the SyncML Data Synchronization Framework is shown by the dotted-line box. The Framework consists 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 might 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 [SYNCDEVDTD] 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 necessary 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 could 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 a SyncML request to archive 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 recipients 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".

5.8.1 Field-level Replace

The SyncML `Replace` command also provides the capability for the originator to send an update to the recipient without having to transfer the entire item. This technique is also called Field-level changes. This feature is extremely useful for the

data types in which relatively concise attributes (for example the "read" status of the e-mail) are more likely to change, than substantially larger attributes like the body of the message or the attachments.

Not all data types are equally suited for being used with Field-level replace. It is the responsibility of the sender to compose the partial items in the corresponding data format in such a manner that they are unambiguously interpreted by the receiver. Also it is the responsibility of the sender to compose the partial items in the corresponding data format ensuring that the format remains valid. If the sender cannot meet these criteria then it **MUST** send a replace for the entire item instead of a field-level replace.

Example:

It is ambiguous to send the field-level change containing the following vCard

```
BEGIN:VCARD
VERSION:2.1
N:Doe;John;;;
TEL;HOME:(321) 654-987
END:VCARD
```

In this case if the receiving side supports more than one HOME phone number, it will have an ambiguity understanding which one was changed.

Example 2:

It is improper to send the field-level change containing the following vCard

```
BEGIN:VCARD
TITLE:Worker
END:VCARD
```

The vCard format mandates the VERSION and N attributes to be present within the item.

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+wxml` identifies the WXML 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.

In addition, the `TargetParent` and `SourceParent` element types MAY be used to provide parent information of the child that is mentioned in the `Target` or `Source` `LocURI` of the sync commands (`Add`, `Replace`, `Move`). Usage of `SourceParent` and `TargetParent` has meaning only when synchronizing objects in a datastore with hierarchical structure.

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`, `Source`, `TargetParent` and `SourceParent` element types within a SyncML document.

Element	Contextual Address Requirement
<code>SyncHdr</code>	
<code>Target</code> and <code>Source</code>	Specifies the address of either the data synchronization server or the client. When addressing the:

	<p>Server - MUST use LocURI element type to specify the unique identifier (e.g. absolute URI form of network address) associated with the server within its domain.</p> <p>Client - MUST use LocURI element type to specify locally unique identifier (e.g. absolute URI form of network address or IMEI URN) associated with the client within its domain.</p>
RespURI	<p>Specifies the address 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 could be performed.</p>
Sync	
Target and Source	<p>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:</p> <p>Server Database - MUST use LocURI element type to specify either the absolute or relative URI for the server database.</p> <p>Client Database - MUST use LocURI element type to specify either the absolute or relative URI for the client database.</p> <p>For a LocURI value, CGI script parameters MAY be appended to the URI to perform selection filtering on the server target.</p>
Search	
Target	<p>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:</p> <p>Server - MUST use LocURI element type to specify the local URI where the search results are to be stored.</p> <p>Client - MUST use LocURI element type to specify either the absolute or relative URI where the search results are to be stored.</p> <p>For a LocURI value, CGI script parameters MAY be appended to the URI to perform selection filtering on the server target.</p>
Source	<p>Specifies one or more addresses on the recipient that are to be searched. When addressing the:</p> <p>Server - MUST use LocURI element type to specify the absolute or relative URI of the databases to be searched.</p> <p>Client - MUST use LocURI element type to specify the absolute or</p>

	relative URI of the databases to be searched.
Map	
Target	<p>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:</p> <p>Server - MUST use LocURI element type to specify the absolute or relative URI of the server database.</p> <p>Client - MUST use LocURI element type to specify the mobile client database.</p>
Source	<p>Specifies the originator database for the Map definition. When addressing the:</p> <p>Server - MUST use LocURI element type to specify the absolute or relative URI of the server database.</p> <p>Client - MUST use LocURI element type to specify the mobile client database.</p>
MapItem	
Target	<p>Specifies the recipient item identifier. When addressing the:</p> <p><i>Server</i> - MUST use LocURI element type to specify the locally unique identifier of the server database item.</p> <p><i>Client</i> - MUST use LocURI element type to specify the locally unique identifier of the mobile client item.</p>
Source	<p>Specifies the originator item identifier. When addressing the:</p> <p><i>Server</i> - MUST use LocURI element type to specify the locally unique identifier of the server database item.</p> <p><i>Client</i> - MUST use LocURI element type to specify the locally unique identifier of the mobile client database item.</p>
Item in an Alert, Exec, Get, Put Commands	
Target and Source	<p>Specifies the address of the database item that is the argument of the SyncML command. When addressing the:</p> <p><i>Server</i> - MUST use LocURI element type to specify the relative URI or URN for the server database.</p> <p><i>Client</i> - MUST use LocURI element type to specify the relative URI of the mobile client database.</p>

	To address individual items within a database using the Get or Put command, URI addressing filtering techniques MUST be used.
Item in Add, Move, Replace Commands	
SourceParent	<p>Specifies parent information of the child that is mentioned in the Source or Target LocURI of the sync command if the objects have hierarchical nature.</p> <p>When addressing the server (client sends to server) – MUST use LocURI element to specify the locally unique identifier for the parent of the client side’s item.</p> <p>When addressing the client (server sends to client) – MUST use LocURI element to specify the temporary global unique identifier for the parent of the server side’s item. The server MUST only send SourceParent if the client’s unique identifier for the parent of the server side’s item is unknown. In all other cases, the server MUST use TargetParent.</p>
TargetParent (<i>used only by the servers</i>)	<p>Specifies parent information of the child that is mentioned in the Target LocURI of the sync command if the objects have hierarchical nature.</p> <p>When addressing the client (server sends to client) – MUST use the LocURI element to specify the client’s unique identifier for the parent of the server side’s item.</p>
Item in Add, Copy, Delete, Move, Replace, Results, Status Commands	
Target and Source	<p>Specifies the address of the database item that is the argument of the SyncML command. When addressing the:</p> <p><i>Server</i> - MUST use LocURI element type to specify the absolute URI, relative URI, URN or locally unique identifier for the server database item.</p> <p><i>Client</i> - MUST use LocURI element type to specify the absolute URI, relative URI, URN or locally unique identifier for the client database item.</p>

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 Data Sync Record and Field Level Filtering

Server data stores frequently contain much more data than can fit into small devices. Other aspects of the protocol enable clients and servers to indicate data store capacity and therefore avoid data overflow conditions, however it is often the case that small devices only want to synchronize a particular, prioritised subset of the data that resides in the server's data store (referred to from this point forth as record filtering). Devices could also allow users to override the level of support for certain properties previously defined in the device info structure (referred to from this point forth as field filtering).

Support for receiving filters **MUST** be indicated in the device info for each data store. Support **MUST** be indicated by the inclusion of the `Filter-Rx` element within the `Datastore` element. The `Filter-Rx` element **MUST** contain a `CTType` and a `VerCT` element. The `CTType` element specifies the filtering grammar supported. For every `Filter-Rx` element a corresponding `FilterCap` element **MUST** be included in the `Datastore` element specifying any keywords or property names that can be filtered on.

A filter is specified by including the `Filter` element for the `Target` of a data store in an `Alert` command. When a `Filter` element is present, the `Filter Meta Type` element **MUST** be included and **MUST** correspond to the mime type the filter applies to. Within the `Filter` element, the `Record`, `Field`, and `FilterType` elements **MAY** be included and all **MAY** be present.

The `Record` element **MUST** contain an `Item` containing a `Meta Type` element representing the filter type used, and one `Data` element representing the query data. The `Data` element **MUST** be a logical expression whereby the expression **MUST** only contain values defined in the `FilterCap` element.

The `Field` element **MUST** contain an `Item` containing a `Meta Type` element representing the device information mime type and one `Data` element containing one or more `Property` elements. The `Property` elements override the corresponding property in the `CTCap` element for the current synchronisation session. Only the properties that differ from the properties specified in the `CTCap` element **MAY** be specified.

The `FilterType` element **MUST** contain a keyword that indicates the type of behavior that the sender is requesting. If the `FilterType` element is not present, then the `FilterType` value of "EXCLUSIVE" **MUST** be assumed.

If an implementation receives a filter record request for a data store that does not support filtering, a status code of 406 (OPTIONAL feature not supported) **MUST** be returned for the command containing the `Filter` element. If a filter record request specifying a filter type that is not supported by the data store is received, a status code 415 (unsupported media type or format) **MUST** be returned for the command containing the `Filter` element. If a filter record request is received which is syntactically incorrect or contains a query that is not supported then a status code of 422 (bad CGI or filter query) **MUST** be returned for the command containing the `Filter` element. If any of those error conditions occur, the sender of the filter **MAY** attempt to resend a new query. If the second query fails as well, a sender **SHOULD** either remove the filter query or terminate the synchronization.

If an implementation received a filter field request for a data store containing properties not previously defined in the corresponding `CTCap` element, then a status code of 400 (bad request) **SHOULD** be returned. Otherwise, the recipient of the filter field request **MUST** override any properties previously retrieved in the `CTCap` element in the device info with the properties present in the filter field request. The properties **MUST** only be overridden for the current synchronization session only.

5.13.1 Filter Behavior Definition

Filtering allows an implementation (most often a client) to constrain the set of items in a data store it wishes to synchronize against and to further constrain the data returned.

The filter only applies to the recipient, that is, an implementation that sends a filter for a synchronisation session is not constrained in the set of items it might send. Filtering serves a different purpose than the `Search` command in that `Sync` using a `Filter` will allow synchronization with a subset of the data in the data store, whereas the `Search` will not.

When a `Filter` has been included within a `Sync` command, the set of data that is defined by the `Filter` **MUST** be fully synchronized during a normal synchronization operation.

If a subsequent `Sync` command is sent (in a subsequent synchronization session) for the same datastore but with a different `Filter` command, the set of data that is defined by the new `Filter` command **MUST** be fully synchronized during a normal synchronization operation. The recipient of the `Filter` command **MUST** no longer send items that were part of the previous filter if those items are no longer part of the new filter.

A recipient **MAY** choose how to insure that this expectation is met. For example, it might require requesting a slow sync, or it might require re-sending records that have been previously synchronized with a different set of fields.

5.13.2 Filter Query Syntax

The filter query is a logical expression contained in the `Filter Record` element and is applied to each item in the recipient's datastore. Often, the values of properties in the data items are compared to literal values supplied by the requestor. Items for which the expression evaluates to true are the set of items for that synchronization session. The filter query is expressed according to a particular grammar. The `Record Item Meta Type` indicates the grammar of the filter query supplied in the `Data` element. This enables the protocol to support additional filter grammars without sacrificing interoperability. The list of grammar types an implementation is capable of receiving **MUST** be indicated through the use of the device info `Filter-Rx` element.

Comparison items **MUST** be valid property names or keywords specified in the `FilterCap` element for the particular filter query grammar being used and all comparison operators **MUST** be supported for each comparison item.. Literal values used in comparisons **MUST** be valid for the property or keyword according to the content type being used for the query. Comparisons are performed using the character encoding specified in the content type, where appropriate.

A grammar **MAY** provide logical operators for conjoining sub-expressions (e.g. AND, OR, NOT) to create arbitrarily complex expressions. A grammar **MAY** provide mechanisms for selecting items based on the presence of properties.

5.13.2.1 Content type requirements

If an implementation supports receiving filters on a given data store, all expressions that test the values of certain base media object properties for that data store, regardless of the query grammar used, are **OPTIONAL**. If the expression is unsupported by the recipient, one of the previously listed status codes **MUST** be returned to the sender. The sender **SHOULD** then modify the expression based on the device info obtained from the recipient. If no expression can be agreed upon between the sender and the recipient then it is up to the sender to determine if the synchronization can be sent without any `Filter` element or if the synchronization **SHOULD** be aborted.

Filtering for all contents types is **OPTIONAL** and **MAY** be supported.

5.13.2.1.1 Contacts Media Object Filter

Filtering for vCard 2.1 [IMVCARD] and vCard 3.0 [RFC2426] objects can be specified using both `Record` and `Field` elements. In the case of `Record` elements, the set of recommended keywords to support are as follows:

```
ct-filter-keyword = "CATEGORIES" | "GROUP"
```

If one chooses to filter based on a property name, some properties like Name ("N") might have several fields. Individual fields could be indicated using a subscript notation. Thus, "N[1]" refers to the family name and "N[2]" refers to the given name.

5.13.2.1.2 Calendar Media Object Filter

Filtering for vCalendar 1.0 [IMVCAL] and iCalendar 2.0 [RFC2445] objects can be specified using both `Record` and `Field` elements. In the case of `Record` elements, the set of recommended keywords to support are as follows:

```
ct-filter-keyword = "SINCE" | "BEFORE" | "STATUS"
```

The format of the "SINCE" and "BEFORE" keywords MUST be a date-time or date format as specified below.

```
date-time      = date "T" time
date           = date-value
date-value     = date-fullyear date-month date-mday
date-fullyear = 4DIGIT
date-month    = 2DIGIT ;01-12
date-mday     = 2DIGIT ;01-28, 01-29, 01-30, 01-31 (based on month/year)
time          = time-hour time-minute time-second time-utc
time-hour     = 2DIGIT           ;00-23
time-minute   = 2DIGIT           ;00-59
time-second   = 2DIGIT           ;00-59
time-utc      = "Z"
```

The "SINCE" keyword represents items that are within or later than the specified date while the "BEFORE" keyword represents items that are earlier than the specified date.

Implementations that support receiving filters for calendar media objects are responsible for expanding recurrence rules ("RRULE" properties) to determine if any instances match the filter conditions. Also, when specifying a date-time format, the use of UTC MUST be used in order to avoid time zone ambiguities. Implementations that cannot provide a UTC date-time value MUST provide a date value instead.

5.13.2.2 CGI Syntax

This section specifies a CGI-like filtering syntax. When using this syntax, the Filter Item Meta Type element MUST be 'syncml:filtertype-cgi'. All implementations that support receiving filter requests MUST support the "syncml:filtertype-cgi" grammar.

The format for the CGI scripting is defined here in an ABNF notation [RFC2234] and the grammar defined here is largely the same as the grammar defined in the SyncML 1.1.2 specification. If the CGI syntax is supported for a data store, then all of the logical CGI scripting primitives in the following table MUST be supported. CGI syntax queries may contain at most 1 type of logical separator, but they MAY contain several logical separators of the same type. For example, they MAY contain several "&OR;" logical operators but a query cannot contain both an "&AND;" and an "&OR;" logical operator in the same expression. The SPACE character MUST be specified by the hexadecimal encoding as stated by the format specification for Uniform Resource Identifiers.

Since the use of lexographic comparison operators is locale specific (for example the use of the "<" operator), devices SHOULD NOT use these operators when specifying a filter for free form text properties. Instead, the following logical operators SHOULD be used in place: "&EQ;", "&iEQ;", "&NE;", "&iNE;", "&CON;", "&iCON;".

Queries using value-based properties (properties that may only contain specific pre-defined values) SHOULD only use the following operators: "&EQ;", "&iEQ;", "&NE;", "&iNE;".

```
VCHAR = %x20-7E ;Visible latin characters within UTF-8 or SPACE character
string-value = 1*VCHAR ;Case sensitive string value
log-equalitycomp = "&EQ;" ;Equal To (case sensitive)
/
                  "&iEQ;" ;Equal To (case insensitive)
/
                  "&NE;" ;Not Equal To (case sensitive)
```

```

/          "&iNE;" ;Not Equal To (case insensitive)

log-op = log-equalitycomp
/ "&GT;"          ;Greater Than (case sensitive)
/ "&iGT;"         ;Greater Than (case insensitive)
/ "&GE;"          ;Greater Than Or Equal To (case sensitive)
/ "&iGE;"         ;Greater Than Or Equal To (case insensitive)
/ "&LT;"          ;Less Than (case sensitive)
/ "&iLT;"         ;Less Than (case insensitive)
/ "&LE;"          ;Less Than Or Equal To (case sensitive)
/ "&iLE;"         ;Less Than Or Equal To (case insensitive)
/ "&CON;"         ;Contains the value (case sensitive)
/ "&iCON;"        ;Contains the value (case insensitive)
/ "&NCON;"        ;Does Not Contain the value (case sensitive)
/ "&iNCON;"       ;Does Not Contain the value (case insensitive)

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

luid-expression = "&LUID;" log-equalitycomp string-value
ct-no-value = "&NULL;"    ; No property value for the item
ct-filter-keyword = string-value    ; Valid content-type specific filter keywords
ct-filter-value = string-value     ; Valid content-type specific property value
ct-expression = ct-filter-keyword (log-op ct-filter-value | log-equalitycomp ct-no-
value)

filter-expression = ct-expression | luaid-expression filter-query = filter-
expression *(log-sep filter-expression)

```

5.13.3 Indicating Filter Support

Implementations that support filtering **MUST** indicate their support in the device info. For each data store, a device indicates the list of filter grammars it is capable of receiving in the `Filter-Rx` element.

5.13.3.1 Minimum Requirements for Filtering support

Implementations that support filtering MUST support receiving the “syncml:filtertype-cgi” grammar. Specifically, they MUST include at least one `Filter-Rx` element specifying the receiving of the “syncml:filtertype-cgi” grammar.

5.13.4 Handling Data Outside Filter Criteria

The following outlines how to handle synchronizing data outside the filter criteria

1. When using an exclusive filter type, the server not only sends all of its changes to the client, but the server MUST also send `Deletes` for all client items that are outside the filter criteria. If the client supports the OPTIONAL `Soft Delete`, the server MAY send `Soft Deletes`; otherwise, the server MUST send `Hard Deletes`.
 - a. First synchronization:
 - i. A two way slow sync has been negotiated with the client sending a filter query
 - ii. The client sends all of its data
 - iii. The server sends all of its data within the filter criteria and `Soft` or `Hard Deletes` for any client data outside the filter criteria
 - b. Subsequent synchronizations:
 - i. A two way sync has been negotiated with the client sending a filter query
 - ii. The client sends all of its changes
 - iii. The server sends all of its changes within the filter criteria and `Hard Deletes` for any client data outside the filter criteria
2. When using an inclusive filter type, before the client sends any of its changes to the server, the client MAY choose to send `Soft Deletes` for all items that were outside the filter criteria after the previous synchronization
 - a. First synchronization:
 - i. A two way slow sync has been negotiated with the client sending a filter query
 - ii. The client sends all of its data
 - iii. The server sends all of its data within the filter criteria
 - iv. After the sync has completed, the client deletes or archives all data outside of the filter query from the client database and marks them so that soft deletes can be sent in the next synchronization
 - b. Subsequent synchronizations:
 - i. A two way sync has been negotiated with the client sending a filter query
 - ii. The client sends `Soft Deletes` for all client data outside of the filter query (that has either been deleted or archived) and also sends all of its changes
 - iii. The server sends all of its changes within the filter criteria
 - iv. After the sync has completed, the client deletes or archives all data outside of the filter query from the client database and marks them so that soft deletes can be sent in the next synchronization. Note that if a client chooses to delete the items outside of the filter query instead of archiving them, it MUST do so immediately after a synchronization in order to prevent the user from modifying the items between synchronizations and thus potentially losing changes made by users.

3. Both the client and server support equivalent filter queries
 - a. First synchronization
 - i. A two way slow sync has been negotiated with the client and server both sending equivalent filter queries yielding identical filter sets
 - ii. The client sends all of its data within the filter criteria
 - iii. The server sends all of its data within the filter criteria
 - b. Subsequent synchronizations
 - i. A two way sync has been negotiated with the client and server both sending equivalent filter queries
 - ii. The client sends all of its changes within the filter criteria
 - iii. The server sends all of its changes within the filter criteria

5.13.5 Examples

The following examples are provided to further illustrate the usage of filtering in SyncML 1.2.

5.13.5.1 Contact Media Objects

5.13.5.1.1 Example 1

In this scenario, the client wishes to sync only Contact items that fall into the “business” or “personal” group.

1. During the initial sync, the client and server exchange their device info.
2. The client analyses the server’s device info, and the client notes that the server supports receiving filters on the Contacts data store for queries using the “syncml:filtertype-cgi” grammar.
 - a. The server includes in its device info the `Filter-Rx` and `FilterCap` elements that it supports.
 - b. The client doesn’t require filtering on any additional fields, so it determines that this server supports the filter it wishes to send.

```

<Datastore>
  <SourceRef>./contacts</SourceRef>
  <DisplayName>Contacts DB</DisplayName>
  ...
  <Filter-Rx>
    <CTType>syncml:filtertype-cgi</CTType>
    <VerCt>1.0</VerCt>
  </Filter-Rx>

```

```

</Filter-Rx>
<CTCap>
  ...
</CTCap>
<FilterCap>
  <CTType>syncml:filtertype-cgi</CTType>
  <VerCt>1.0</VerCt>
  <FilterKeyword>GROUP</FilterKeyword>
  <PropName>CATEGORIES</PropName>
</FilterCap>
</Datastore>

```

3. The client sends an Alert for the Contacts data store with a filter.
 - a. It includes the Filter Meta Type element to indicate the content type desired (vCard in this example).
 - b. It includes a Filter Record element with a Meta Type value of “syncml:filtertype-cgi” to indicate the grammar being used.
 - c. The filter query in the Item Data element contains a value of “GROUP&iCON;business&OR; GROUP &iCON;personal” to constrain the items synchronized to those that fall into the “business” or “personal” group (case insensitive).

```

<Alert>
  <Data>200</Data>
  <Item>
    <Target>
      <LocURI>./contacts</LocURI>

      <Filter>
        <Meta><Type>text/x-vcard</Type></Meta>
      <Record>
        <Item>
          <Meta><Type>syncml:filtertype-cgi</Type></Meta>
          <Data> GROUP&iCON;business&OR;GROUP&iCON;personal</Data>
        </Item>
      </Record>
    </Filter>
  </Item>

```

```

</Target>

<Source>
  <LocURI>dev-contacts</LocURI>
</Source>
</Item>
</Alert>

```

4. The server receives the `Alert` with the `Filter Record` element.
 - a. It determines that it supports the filter operation for the data store, content type, filter grammar, and properties.
 - b. It replies with a status code of 200 for the `Alert`, indicating that it can satisfy the request to sync with filtering.
5. The synchronization process continues normally.
 - a. The client sends all of its changes to the server (the filter constraint is not imposed on it in this scenario).
 - b. The server sends changes only for items that satisfy the filter query.

5.13.5.1.2 Example 2

In this scenario, the client wishes to sync only `Contact` items that fall into the “business” or “personal” group. Additionally the client has indicated in its device info that it supports the `PHOTO` property, but it does not wish to receive the `PHOTO` property from the server for this synchronization request.

1. During the initial sync, the client and server exchange their device info.
2. The client analyses the server’s device info, and the client notes that the server supports receiving filters on the `Contacts` data store for queries using the “`syncml:filtertype-cgi`” grammar.
 - a. The server includes in its device info the `Filter-Rx` and `FilterCap` elements that it supports.
 - b. The client doesn’t require filtering on any additional fields, so it determines that this server supports the filter it wishes to send.
3. The client sends an `Alert` for the `Contacts` data store with a filter.
 - a. It includes the `Filter Meta Type` element to indicate the content type desired (vCard in this example).
 - b. It includes a `Filter Record` element with a `Meta Type` value of “`syncml:filtertype-cgi`” to indicate the grammar being used.
 - c. The filter query in the `Item Data` element contains a value of “`GROUP&iCON;business&OR;GROUP&iCON;personal`” to constrain the items synchronized to those that fall into the “business” or “personal” group (case insensitive).
 - d. It includes a `Filter Field` element containing a `Property` element set to “`PHOTO`” containing a `MaxSize` element set to 0 (zero).

```

<Alert>
  <Data>200</Data>

```

```

<Item>
  <Target>
    <LocURI>./contacts</LocURI>

    <Filter>
      <Meta><Type>text/x-vcard</Type></Meta>
      <Record>
        <Item>
          <Meta><Type>syncml:filtertype-cgi</Type></Meta>
          <Data> GROUP&iCON;business&OR;GROUP&iCON;personal</Data>
        </Item>
      </Record>
      <Field>
        <Item>
          <Meta><Type>application/vnd.syncml-devinf+xml</Type></Meta>
          <Data>
            <Property>
              <PropName>PHOTO</PropName>
              <MaxSize>0</MaxSize>
              <NoTruncate/>
            </Property>
          </Data>
        </Item>
      </Field>
    </Filter>
  </Target>
  <Source>
    <LocURI>dev-contacts</LocURI>
  </Source>
</Item>
</Alert>

```

4. The server receives the Alert with the Filter Record and Field elements.

- a. It determines that it supports the filter operations for the data store, content type, filter grammar, and properties.
 - b. It replies with a status code of 200 for the Alert, indicating that it can satisfy the request to sync with filtering.
5. The synchronization process continues normally.
- a. The client sends all of its changes to the server (the filter constraint is not imposed on it in this scenario).
 - b. The server sends changes only for items that satisfy the filter query. The server does not send any PHOTO properties since the client has requested that it wishes to receive only 0 bytes of this property for this synchronization request and the value should SHOULD not be truncated.

5.13.5.2 Calendar Media Objects

5.13.5.2.1 Example 1

In this scenario, the client wishes to synchronize calendar items that fall within a two week window of time (starting with the current date).

1. During the initial sync, the client and server exchange their device info.
2. The client analyses the server's device info, and the client notes that the server supports receiving filters on the Calendar data store for queries using the "syncml:filtertype-cgi" grammar.
 - a. The server includes in its device info the `Filter-Rx` and `FilterCap` elements that it supports. This includes the SINCE and BEFORE keywords.
 - b. The client doesn't require filtering on any additional fields, so it determines that this server supports the filter it wishes to send.

```

<Datastore>
  <SourceRef>./calendar/events</SourceRef>
  <DisplayName>Calendar Agenda DB</DisplayName>
  ...
  <Filter-Rx>
    <CTType>syncml:filtertype-cgi</CTType>
    <VerCt>1.0</VerCt>
  </Filter-Rx>
  <CTCap>
    ...
  </CTCap>
  <FilterCap>
    <CTType>syncml:filtertype-cgi</CTType>
    <VerCt>1.0</VerCt>

```

```

    <FilterKeyword>BEFORE</FilterKeyword>

    <FilterKeyword>SINCE</FilterKeyword>

  </FilterCap>
</Datastore>

```

3. The client sends an `Alert` for the Calendar data store with a filter.
 - a. It includes the `Filter Meta Type` element to indicate the content type desired (iCalendar in this example).
 - b. It includes a `Filter Record` element with a `Meta Type` value of “syncml:filtertype-cgi” to indicate the grammar being used.
 - c. The filter query in the `Item Data` element contains a value of “SINCE&EQ;20030606T000000Z&AND;BEFORE&EQ;20030620T000000Z” to constrain the items synchronized to those that occur between June 6, 2003 and June 19, 2003 inclusive, using the “SINCE” and “BEFORE” keywords.

```

<Alert>
  <Data>200</Data>
  <Item>
    <Target>
      <LocURI>./calendar/events</LocURI>

      <Filter>
        <Meta><Type>text/x-vcalendar</Type></Meta>

        <Record>
          <Item>
            <Meta><Type>syncml:filtertype-cgi</Type></Meta>

            <Data>SINCE&EQ;20020707T000000Z&AND;BEFORE&EQ;20020728T000000Z</Data>
          </Item>
        </Record>
      </Filter>
    </Target>
    <Source>
      <LocURI>dev-agenda</LocURI>
    </Source>
  </Item>
</Alert>

```

4. The server receives the `Alert` with the `Filter Record` element.
 - a. It determines that it supports the filter operation for the data store, content type, filter grammar, and properties.
 - b. It replies with a status code of 200 for the `Alert`, indicating that it can satisfy the request to sync with filtering.
5. The synchronization process continues normally.
 - a. The client sends all of its changes to the server (the filter constraint is not imposed on it in this scenario).
 - b. The server sends changes only for items that satisfy the filter query.

5.13.5.2.2 Example 2

In this scenario, the client wishes to synchronize task items (iCalendar vTODO components) that have not been completed and are due within the next 2 weeks (using the `DUE` iCalendar property). The client also wishes to limit the size of the `DESCRIPTION` property to 100 bytes and only receive `ATTACH` properties that are less than 1000 bytes.

1. During the initial sync, the client and server exchange their device info.
2. The client analyses the server's device info, and the client notes that the server supports receiving filters on the Calendar data store for queries using the "syncml:filtertype-cgi" grammar.
 - a. The server includes in its device info the `Filter-Rx` and `FilterCap` elements that it supports. This includes the `STATUS` and `DUE` keywords.
 - b. The client doesn't require filtering on any additional fields, so it determines that this server supports the filter it wishes to send.

```

<Datastore>
  <SourceRef>./calendar/tasks</SourceRef>
  <DisplayName>Task DB</DisplayName>
  ...
  <Filter-Rx>
    <CTType>syncml:filtertype-cgi</CTType>
    <VerCt>1.0</VerCt>
  </Filter-Rx>
  <CTCap>
    ...
  </CTCap>
  <FilterCap>
    <CTType>syncml:filtertype-cgi</CTType>
    <VerCt>1.0</VerCt>
    <PropName>DUE</PropName>
    <PropName>STATUS</PropName>
  </FilterCap>

```



```

</FilterCap>
</Datastore>

```

3. The client sends an `Alert` for the `Calendar` data store with a filter.
 - a. It includes the `Filter Meta Type` element to indicate the content type desired (`iCalendar` in this example).
 - b. It includes a `Filter Record` element with a `Meta Type` value of “`syncml:filtertype-cgi`” to indicate the grammar being used.
 - c. The filter query in the `Item Data` element contains a value of “`DUE&LE;20030620T000000Z&AND;STATUS&NE;COMPLETED`” to constrain the items synchronized to those that are due before June 20, 2003 and have not been completed.
 - d. It includes a `Filter Field` element containing a `Property` element set to “`DESCRIPTION`” containing a `MaxSize` element set to 100 and a second property element set to “`ATTACH`” containing a `MaxSize` of 1000 with the `NoTruncate` tag present.

```

<Alert>
  <Data>200</Data>
  <Item>
    <Target>
      <LocURI>./calendar/tasks</LocURI>
      <Meta><Type>text/x-vcalendar</Type></Meta>
    <Filter>
      <Meta><Type>text/x-vcalendar</Type></Meta>
    <Record>
      <Item>
        <Meta><Type>syncml:filtertype-cgi</Type></Meta>
        <Data>DUE&LE;20030728T000000Z&AND;STATUS&NE;COMPLETED</Data>
      </Item>
    </Record>
    <Field>
      <Item>
        <Meta><Type>application/vnd.syncml-devinf+xml</Type></Meta>
        <Data>
          <Property>
            <PropName>DESCRIPTION</PropName>
            <MaxSize>100</MaxSize>
          </Property>

```

```
<Property>
  <PropName>ATTACH</PropName>
  <MaxSize>1000</MaxSize>
  </NoTruncate>
</Property>
</Data>
</Item>
</Field>
</Filter>
</Target>
<Source>
  <LocURI>dev-tasks</LocURI>
</Source>
</Item>
</Alert>
```

4. The server receives the `Alert` with the `Filter Record` element.
 - a. It determines that it supports the filter operation for the data store, content type, filter grammar, and properties.
 - b. It replies with a status code of 200 for the `Alert`, indicating that it can satisfy the request to sync with filtering.
5. The synchronization process continues normally.
 - a. The client sends all of its changes to the server (the filter constraint is not imposed on it in this scenario).
 - b. The server sends changes only for items that satisfy the filter query. The server has truncated the `DESCRIPTION` properties to 100 bytes and has also not sent any `ATTACH` properties that are larger than 1000 bytes.

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.

The formal static conformance requirements for the SyncML data description elements are found in **Static Conformance Requirements (Normative)**

The following table further elucidates 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
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
Field	MAY	MAY	MAY	MAY
Filter	MAY	MAY	MAY	MAY
FilterType	MAY	MAY	MAY	MAY
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
Record	MAY	MAY	MAY	MAY
RespURI	MAY	MUST	MAY	MUST
SessionID*	MUST	MUST	MUST	MUST
SftDel	MAY	MAY	MAY	MAY
Source	MUST	MUST	MUST	MUST
SourceParent	MAY	MAY	MAY	MAY
SourceRef	MUST	MUST	MUST	MUST
Target	MUST	MUST	MUST	MUST
TargetParent**	MAY	MUST NOT	MUST NOT	MAY
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.

**TargetParent is never sent by a client

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' >ZG9iZWVhdmUNCg==</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 Field

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

Example: : The following is an example of a `Field` element used within a `Filter` element to define the characteristics of the subset of data to be synchronized. The `Filter` `Field` element contains a `Property` element set to “PHOTO” containing a `MaxSize` element set to 0 (zero). This indicates to the server that it SHOULD NOT send any PHOTO properties since the client has requested that it wishes to receive only 0 bytes of this property for this synchronization request and the value SHOULD NOT be truncated.

```
<Filter>
.....
  <Field>
    <Item>
      <Meta><Type>application/vnd.syncml-devinf+xml</Type></Meta>
      <Data>
        <Property>
          <PropName>PHOTO</PropName>
          <MaxSize>0</MaxSize>
          <NoTruncate/>
        </Property>
      </Data>
    </Item>
  </Field>
.....
</Filter/>
```

6.1.8 Filter

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

Example: The following is an example of a `Filter` element which

1. uses the `Record` element with a `Meta` `Type` value of “syncml:filtertype-cgi” to indicate the grammar being used.

- uses the Item Data element to constrain the items synchronized to those that fall into the “business” or “personal” group (case insensitive) with the cgi expression “GROUP&iCON;business&OR; GROUP &iCON;personal”.

```
<Filter>
  <Record>
    <Item>
      <Meta><Type>syncml:filtertype-cgi</Type></Meta>
      <Data>GROUP&iCON;business&OR;GROUP&iCON;personal</Data>
    </Item>
  </Record>
</Filter>
```

6.1.9 FilterType

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

Example: The following is an example of a Filter element which uses the EXCLUSIVE FilterType keyword to indicate that

- the sender is requesting that the set of data items to be synchronized MUST be exactly the set of items specified by the Filter.
- Additional items in the data store MUST not be synchronized and the recipient of the Filter MUST request that additional items be removed from the sender device.

```
<Filter>
  <FilterType>EXCLUSIVE</FilterType>
  <Record>
    <Item>
      <Meta><Type>syncml:filtertype-cgi</Type></Meta>
      <Data>GROUP&iCON;business&OR;GROUP&iCON;personal</Data>
    </Item>
  </Record>
</Filter>
```

6.1.10 Final

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

Example:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>...blah, blah...</SyncHdr>
  <SyncBody>
    ...blah, blah...
    <Final/>
  </SyncBody>
</SyncML>
```


6.1.11 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></Source>
  </Item>
</Get>
```

6.1.12 LocName

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

6.1.13 LocURI

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

Example:

```
<SyncHdr>
  <VerDTD>1.2</VerDTD>
  <VerProto>SyncML/1.2</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.datasync.org/servlet/syncit/</LocURI></Target>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>
```

6.1.14 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>

```

6.1.15 MsgID

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

Example:

```

<SyncHdr>
  <VerDTD>1.2</VerDTD>
  <VerProto>1.2</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target><LocURI>http://www.syncml.host.com/</LocURI></Target>
  <Source><LocURI>IMEI:001004FF1234567</LocURI></Source>
</SyncHdr>

```

6.1.16 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>

```

6.1.17 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>
```

6.1.18 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>
```

6.1.19 NumberOfChanges

Restrictions: The element type MUST 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>` MUST 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>

```

6.1.20 Record

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

Example: The following is an example of a `Record` element used within a `Filter` element to define the characteristics of the subset of data to be synchronized.

1. The `Meta Type` value of “syncml:filtertype-cgi” indicates the grammar being used.
2. The `Item Data` element constrains the items synchronized to those that fall into the “business” or “personal” group (case insensitive) with the cgi expression “GROUP&iCON;business&OR; GROUP &iCON;personal”.

```

<Filter>
  <Record>
    <Item>
      <Meta><Type>syncml:filtertype-cgi</Type></Meta>
      <Data>GROUP&iCON;business&OR;GROUP&iCON;personal</Data>
    </Item>
  </Record>
</Filter>

```

6.1.21 RespURI

Restrictions: CGI scripting can be used to convey "hints" to the recipient about when a response could 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>1.2</VerDTD>
  <VerProto>SyncML/1.2</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>

```

6.1.22 SessionID

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

Example:

```

<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>SyncML/1.2</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>

```

6.1.23 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>

```

6.1.24 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>1.2</VerDTD>
  <VerProto>SyncML/1.2</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>
```

6.1.25 SourceParent

Restrictions:

`SourceParent` provides parent information of the child that is mentioned in the `Source LocURI` or `Target LocURI` of the sync commands such as `Move`, `Add`, `Replace`. `SourceParent` MUST be specified in `Add`, `Replace`, `Move`, if and only if the objects have hierarchical nature, i.e. have a parent and child relation. `SourceParent` has meaning only when synchronizing objects in a datastore with hierarchical structure.

In case the parent container is `root` then the value of the `SourceParent` MUST be indicated by `'/'`, without the quotes.

The semantics of the `SourceParent` element differs depending on who the sending side is:

- ❑ If client sends the `SourceParent` element then it represents the client side's ID of an item.
- ❑ If server sends the `SourceParent` element then it represents a temporary ID (GUID) of the item that was previously sent by the server to the client and for which the `Map` was not yet received by the server.

In the case of moving a child on the server that was already synced with the client to a new parent, which hasn't been synced, the server MUST use `SourceParent`. In such situations the new parent has to be added before the `Move` can be performed. This scenario is further illustrated by the example

Example:

```
<Add>
  <CmdID>12345</CmdID>
  <Item>
    <Source><LocURI>1002345</LocURI></Source>

    <Data>
      ...
    </Data>
  </Item>
</Add>

<!-- Since this is an add from the server to the client, client will
assign an id on its own to the folder added by the server. The mapping of
the client's id with server TempGUID '1002345' MUST be maintained by the
```

```

client till the end of Package 4 -->
<Move>
  <CmdID>1234</CmdID>
  <Meta><Type xmlns='syncml:metinf'>text/plain</Type></Meta>
  <Item>
    <Target><LocURI>110</LocURI></Target>
    <SourceParent><LocURI>1002345</LocURI></SourceParent>
  </Item>
</Move>

<!-- Since server hasn't received the mapping yet, server will use the
same TempGUID '1002345' when addressing the NEW Parent folder in the
SourceParent of the Move Command -->

```

6.1.26 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>

```

6.1.27 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>1.2</VerDTD>
  <VerProto>SyncML/1.2</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>

```

6.1.28 TargetParent

Restrictions:

TargetParent provides parent information of the child that is mentioned in the Target LocURI or Source LocURI of the sync commands such as Add, Move and Replace. TargetParent MUST be specified in Add, Replace and Move, if and only if the objects have hierarchical nature, i.e. have a parent and child relation. Usage of TargetParent has no meaning if objects have a flat structure. TargetParent has meaning only when synchronizing objects with hierarchical structure.

In case the parent container is a root then the value of the TargetParent MUST be indicated by '/', without the quotes.

`TargetParent` element always represents a LUID, i.e. the ID of client that was previously sent by the client using `Map` operation. `TargetParent` MUST be used by servers only.

Example: the server requests the client to move the item having the LUID '110'.

```
<Move>
  <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/plain</Type></Meta>
  <Item>
    <Target><LocURI>110</LocURI></Target>
    <TargetParent><LocURI>1234</LocURI></TargetParent>
  </Item>
</Move>
```

6.1.29 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>
```

6.1.30 VerDTD

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

Example:

```
<SyncHdr>
  <VerDTD>1.2</VerDTD>
  <VerProto>SyncML/1.2</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>
    
```

6.1.31 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 1.2.

Example:

```

<SyncHdr>
  <VerDTD>1.2</VerDTD>
  <VerProto>SyncML/1.2</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.

The formal static conformance requirements for the SyncML data description elements are found in **Static Conformance Requirements (Normative)**

The following table further elucidates 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
SyncML	MUST	MUST	MUST	MUST
SyncHdr	MUST	MUST	MUST	MUST
SyncBody	MUST	MUST	MUST	MUST

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:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>SyncML/1.2</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:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</Version>
    <VerProto>SyncML/1.2</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:SYNCML1.2' >
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>SyncML/1.2</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>./devinf12</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.

The formal static conformance requirements for the SyncML data description elements are found in **Static Conformance Requirements (Normative)**

The following table further elucidates 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
------	------	------	------	------

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 an Item with data that does contain 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>IBM</Man>
      <Mod>WorkPad</Mod>

      <DevTyp>pda</DevTyp>
      <DevID>J. Smith</DevID>
      <FwV>PalmOSv3.0</FwV>
      <OEM>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>IBM</Man>
      <Mod>WorkPad</Mod>

      <DevTyp>pda</DevTyp>
      <DevID>J. Smith</DevID>
      <FwV>PalmOSv3.0</FwV>
      <OEM>Palm, Inc.</OEM>
    </DevInf>
  </Data>
</Meta>
    
```

6.4 Protocol Management Elements

The formal static conformance requirements for the SyncML protocol management elements are found in Static Conformance Requirements (Normative)

The following table further elucidates the static conformance requirements for the SyncML protocol management elements for devices conforming to this specification.

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

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

The formal static conformance requirements for the SyncML protocol command elements are found in **Static Conformance Requirements** (Normative)

The following table further elucidates 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
Move	MAY	MAY	MAY	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.

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.

When synchronizing hierarchical objects, `Add` command `Item` element **MUST** include parent information. For this purpose `SourceParent` or `TargetParent` **MUST** be used by the sending device referring to an existing parent. Refer to each of the respective sections in this specification to learn more about when to use `SourceParent` and `TargetParent`.

Example: The client requests the server to add a new item to the root.

```
<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>
</Item>
```

```

<Source><LocURI>1002345</LocURI></Source>

<SourceParent><LocURI>/</LocURI></SourceParent>

<Data>
  ...
<Data>
</Item>
</Add>

```

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 exist 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.

When synchronizing hierarchical objects, `Move` or `Delete` command **MUST** be issued for all children before issuing a `Delete` for the parent. In case a `Delete` is received for the parent containing child items, an exception status code 427 "Item not empty", **MAY** be returned.

Example:

```
<Delete>
  <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>
    <Source><LocURI>1002345</LocURI></Source>

  </Item>
</Delete>
```

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 could 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>Zz6EivR3yaaaENcRN6lpAQ==</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 necessary 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 Move

Move command allows moving items (ex: files, folder, emails, vcards) from current location to a new location. This command MUST only 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.

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.

Move command MUST include parent information, either SourceParent or TargetParent element within Item. The usage of SourceParent and TargetParent within Move command depends on who is requesting the Move operation, client or server. Refer to each of the respective sections in this specification to learn more about when to use SourceParent and TargetParent.

One or more Item element types MUST be specified. Even if Data is specified in Item, it will be ignored by the recipient. Thus, Move is strictly used for moving items and thus in situations where items could get modified and moved, Move MUST NOT be used. 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.

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

If the recipient determines that the data item couldn't be moved on recipient's database, then the (428) Move Failed 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.

Example: The following is an example of a Move command sent by the server. The server requests the client to move the item which has the LUID '110'.

```

<Move>
  <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/plain</Type></Meta>
  <Item>
    <Target><LocURI>110</LocURI></Target>
    <TargetParent><LocURI>1234</LocURI></TargetParent>
  </Item>
</Move>

```

Example: The item '111' has been moved from the folder '1212' to the folder '1313' in the client's datastore and both parents (i.e. folders) have earlier been synchronized with the server. The client creates the following Move command:

```

<Move>
  <CmdID>2</CmdID>
  <Meta><Type xmlns='syncml:metinf'>application/vnd.omads-
file</Type></Meta>
  <Item>
    <Source><LocURI>111</LocURI></Target>
    <SourceParent><LocURI>1313</LocURI></SourceParent>
  </Item>
</Move>

```

6.5.11 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>

  <Meta><Type xmlns='syncml:metinf'>application/vnd.syncml-
devinf+xml</Type></Meta>
  <Item>
    <Source><LocURI>./devinf12 </LocURI></Source>
    <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>

```

6.5.12 Replace

Restrictions: The `Replace` command is used to replace data on the recipient. This command **MUST** only be specified within a `Sync` command. The `Replace` may be partial(field-level `Replace`) or full. 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. If the `Replace` is partial, then the `FieldLevel` `Metatag` **MUST** be used. The sender **MAY** include unchanged fields of the item inside `<Data>` element if the `<FieldLevel>` element is included in the `<Meta>` element. The receiver **MUST NOT** remove any fields of the item in the database that are not present in `<Data>` element content if the `<FieldLevel>` element is included in `<Meta>` element. If the `<FieldLevel>` is included in `<Meta>`, the partial item inside the `<Data>` **MUST** still satisfy the validity rules defined for the content type specified in `<Type>` element. The scope of the meta-information is limited to the command.

Example:

```

<Replace>
  <CmdID>3</CmdID>

  <Meta>
    <FieldLevel xmlns="syncml:metinf"/>
    <Type xmlns="syncml:metinf">x-type/x-subtype</Type>
  </Meta>

  <Item>
    <Target>
      <LocURI>244</LocURI>
    </Target>
    <Data>
    ...

```

```

    </Data>
  </Item>
</Replace>

```

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.

When synchronizing hierarchical objects, the `Replace` command MUST include parent information. For this purpose `SourceParent` or `TargetParent` MUST be used by sending device referring to an existing parent.. Refer to each of the respective sections in this specification to learn more about when to use `SourceParent` and `TargetParent`.

In the case where an item has been modified and moved, `Replace` MUST be used instead of `Move`.

If `Replace` command is used by client for add operation, then client MUST add parent first before adding all children of that parent. For example, if an item B has a parent A, then before adding item B, parent A MUST be added by the client.

Example:

```

<Replace>
  <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>
    <Source><LocURI>1002345</LocURI></Source>

    <SourceParent><LocURI>/</LocURI></SourceParent>

    <Data>
      ...
    </Data>
  </Item>
</Replace>

```

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.

In case the receiver is unable to process a partial item update (e.g. when the item does not exist on receiver), it MUST return status code 426 (Partial item not accepted).

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>
```

6.5.13 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>./TELECOM/PB</TARGETREF>
  <SOURCEREF>HTTP://WWW.DATASYNC.COM/SERVLET/</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>

```

6.5.14 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>
```

6.5.15 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>
```

6.5.16 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.

The `Target` element type **MUST** be used to specify the recipient database to be synchronized.

The `Source` element type **MUST** be used to specify 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
<i>Alert Codes used for user alerts</i>		
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.
<i>Alert Codes used at the synchronization initialization</i>		
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.
<i>Alert Codes used by the server when alerting the synchronization</i>		
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.
<i>Special Alert Codes</i>		
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	SUSPEND	Suspend synchronization session
225	RESUME	Resume synchronization session
226-250	-	Reserved for future SyncML usage.

8. Base Media and Content formats

Data Type	Media Type	URI	Content Format
Contact	text/x-vcard	http://imc.org/pdi/vcard-21.doc	vCard 2.1
	text/vcard	http://www.ietf.org/rfc/rfc2426.txt	vCard 3.0
Calendar	text/x-vcalendar	http://www.imc.org/pdi/vcal-10.doc	vCalendar 1.0
	text/calendar	http://www.ietf.org/rfc/rfc2445.txt	iCalendar 2.0
Memos	text/plain	http://www.ietf.org/rfc/rfc2046.txt	
Tasks	text/x-vcalendar	http://www.imc.org/pdi/vcal-10.doc ,	vCalendar 1.0
	text/calendar	http://www.ietf.org/rfc/rfc2445.txt	iCalendar 2.0
Email	message/rfc822	http://www.ietf.org/rfc.html	RFC822
			RFC2822
			RFC2045
	application/vnd.omads-email	http://www.openmobilealliance.org/	XML object
File	application/vnd.omads-file	http://www.openmobilealliance.org/	XML object
Folder	application/vnd.omads-folder	http://www.openmobilealliance.org/	XML object

9. MIME Media Type Registration

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

9.1 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, verstd. 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 "1.2" 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"

vertdtd 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 "1.2" is to be assumed.

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

```
vertdtd-param = ";" "vertdtd" "=" 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 could 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](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): XXML

Person & email address to contact for further information: admins@syncml.org

Intended usage: COMMON

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

9.2 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 "1.2" 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"
```

vertdtd 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 "1.2" is to be assumed.

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

```
vertdtd-param = ";" "vertdtd" "=" 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 could 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 and MANDATORY within implementations conforming to this specification.

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

A.1 Client Data Sync Usage of SyncML Representation

Table 1 – Client Common Use Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-CUE-C-001	Support for Archive element	6.1.1	O	
SCR-DS-CUE-C-002	Support for Chal element	6.1.2	M	SCR-DS-DDE-C-003
SCR-DS-CUE-C-003	Support for Cmd element	6.1.3	M	
SCR-DS-CUE-C-004	Support for CmdId element	6.1.4	M	
SCR-DS-CUE-C-005	Support for CmdRef element	6.1.5	M	
SCR-DS-CUE-C-006	Support for Cred element	6.1.6	M	SCR-DS-DDE-C-001
SCR-DS-CUE-C-007	Support for Field element	6.1.7	O	
SCR-DS-CUE-C-008	Support for Filter element	6.1.8	O	SCR-DS-CUE-C-09 AND (SCR-DS-CUE-C-007 OR SCR-DS-CUE-C-020)
SCR-DS-CUE-C-009	Support for FilterType element	6.1.9	O	
SCR-DS-CUE-C-010	Support for Final element	6.1.10	M	
SCR-DS-CUE-C-011	Support for Lang element	6.1.11	O	
SCR-DS-CUE-C-012	Support for LocName element	6.1.12	O	
SCR-DS-CUE-C-013	Support for LocURI element	6.1.13	M	
SCR-DS-CUE-C-014	Support for MoreData element	6.1.14	O	
SCR-DS-CUE-C-015	Support for MsgID element	6.1.15	M	
SCR-DS-CUE-C-016	Support for MsgRef element	6.1.16	M	
SCR-DS-CUE-C-017	Support for NoResp element	6.1.17	M	
SCR-DS-CUE-C-018	Support for NoResults element	6.1.18	O	
SCR-DS-CUE-C-019	Support for NumberOfChanges element	6.1.19	O	
SCR-DS-CUE-C-020	Support for Record element	6.1.20	O	

SCR-DS-CUE-C-021	Support for RespURI element	6.1.21	M	
SCR-DS-CUE-C-022	Support for SessionID element	6.1.22	M	
SCR-DS-CUE-C-023	Support for SftDel element	6.1.23	O	
SCR-DS-CUE-C-024	Support for Source element	6.1.24	M	SCR-DS-CUE-C-013
SCR-DS-CUE-C-025	Support for SourceParent	6.1.25	O	SCR-DS-CLIENT-012
SCR-DS-CUE-C-026	Support for SourceRef element	6.1.26	M	
SCR-DS-CUE-C-027	Support for Target element	6.1.27	M	SCR-DS-CUE-C-013
SCR-DS-CUE-C-028	Support for TargetParent	6.1.28	O	SCR-DS-CLIENT-012
SCR-DS-CUE-C-029	Support for TargetRef element	6.1.29	M	
SCR-DS-CUE-C-030	Support for VerDTD element	6.1.30	M	
SCR-DS-CUE-C-031	Support for VerProto element	6.1.31	M	

Table 2 – Client Message Container Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-MCE-C-001	Support for SyncML element	6.2.1	M	SCR-DS-MCE-C-002 AND SCR-DS-MCE-C-003
SCR-DS-MCE-C-002	Support for SyncHdr element	6.2.2	M	SCR-DS-CUE -C-015 AND SCR-DS-CUE -C-022 AND SCR-DS-CUE-C-024 AND SCR-DS-CUE-C-027 AND SCR-DS-CUE-C-030 AND SCR-DS-CUE-C-031
SCR-DS-MCE-C-003	Support for SyncBody element	6.2.3	M	SCR-DS-PCE-C-001 AND SCR-DS-PCE-C-002 AND SCR-DS-PCE-C-005 AND SCR-DS-PCE-C-007 AND SCR-DS-PCE-C-010 AND SCR-DS-PCE-C-011 AND SCR-DS-PCE-C-015 AND SCR-DS-PCE-C-016

Table 3 – Client Data Description Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-DDE-C-001	Support for Data element	6.3.1	M	
SCR-DS-DDE-C-002	Support for Item element	6.3.2	M	
SCR-DS-DDE-C-003	Support for Meta element	6.3.3	M	SCR-DSDM-METINF-S-011

Table 4 – Client Protocol Management Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-PME-C-001	Support for Status element	6.4.1	M	SCR-DS-CUE-C-003 AND SCR-DS-CUE-C-004 AND SCR-DS-CUE-C-005 AND SCR-DS-CUE-C-015 AND SCR-DS-CUE-C-024 AND SCR-DS-CUE-C-026 AND SCR-DS-DDE-C-001 AND SCR-DS-DDE-C-002

Table 5 – Client Protocol Command Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-PCE-C-001	Support for Add element	6.5.1	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-002	Support for Alert element	6.5.2	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-003	Support for Atomic element	6.5.3	O	SCR-DS-CUE-C-004
SCR-DS-PCE-C-004	Support for Copy element	6.5.4	O	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-005	Support for Delete element	6.5.5	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-006	Support for Exec element	6.5.6	O	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-007	Support for Get element	6.5.7	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-008	Support for Map element	6.5.8	O	SCR-DS-CUE-C-004 AND SCR-DS-PCE-C-009
SCR-DS-PCE-C-009	Support for MapItem element	6.5.9	O	SCR-DS-CUE-C-024 AND SCR-DS-CUE-C-027
SCR-DS-PCE-C-010	Support for Move element	6.5.10	O	
SCR-DS-PCE-C-010	Support for Put element	6.5.10	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-011	Support for Replace element	6.5.11	M	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002

SCR-DS-PCE-C-012	Support for Result element	6.5.12	O	SCR-DS-CUE-C-004 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-013	Support for Search element	6.5.13	O	SCR-DS-CUE-C-004 AND SCR-DS-CUE-C-024 AND SCR-DS-DDE-C-002 AND SCR-DS-DDE-C-003
SCR-DS-PCE-C-014	Support for Sequence element	6.5.14	O	SCR-DS-CUE-C-004
SCR-DS-PCE-C-015	Support for Status element	6.4.1	M	SCR-DS-CUE-C-003 AND SCR-DS-CUE-C-004 AND SCR-DS-CUE-C-005 AND SCR-DS-CUE-C-016 AND SCR-DS-CUE-C-026 AND SCR-DS-CUE-C-029 AND SCR-DS-DDE-C-001 AND SCR-DS-DDE-C-002
SCR-DS-PCE-C-016	Support for Sync element	6.5.15	M	SCR-DS-CUE-C-004

Table 6 – Client Content Formats

Item	Functionality	Reference	Status	Requirement
SCR-DS-CONTENT-C-001	Support for Contact Synchronization		O	SCR-DS-CONTENT-C-008 OR SCR-DS-CONTENT-C-009
SCR-DS-CONTENT-C-002	Support for Calendar Synchronization		O	SCR-DS-CONTENT-C-010 OR SCR-DS-CONTENT-C-011
SCR-DS-CONTENT-C-003	Support for Memo Synchronization		O	SCR-DS-CONTENT-C-012
SCR-DS-CONTENT-C-004	Support for Task Synchronization		O	SCR-DS-CONTENT-C-013
SCR-DS-CONTENT-C-005	Support for Email Synchronization		O	SCR-DS-CONTENT-C-014 OR SCR-DS-CONTENT-C-015 OR SCR-DS-CONTENT-C-016 OR SCR-DS-CONTENT-C-017
SCR-DS-CONTENT-C-006	Support for File Synchronization		O	SCR-DS-CONTENT-C-018
SCR-DS-CONTENT-C-007	Support for Folder Synchronization		O	SCR-DS-CONTENT-C-019
SCR-DS-CONTENT-C-008	Support for vCard 2.1		O	
SCR-DS-CONTENT-C-009	Support for vCard 3.0		O	
SCR-DS-CONTENT-C-010	Support for vCalendar 1.0		O	
SCR-DS-CONTENT-C-011	Support for iCalendar 2.0		O	

SCR-DS-CONTENT-C-012	Support for text/plain		O	
SCR-DS-CONTENT-C-013	Support for vTodo 1.0		O	
SCR-DS-CONTENT-C-014	Support for message/rfc822		O	
SCR-DS-CONTENT-C-015	Support for message/rfc2822		O	
SCR-DS-CONTENT-C-016	Support for Message/rfc2045		O	
SCR-DS-CONTENT-C-017	Support for x-email		O	
SCR-DS-CONTENT-C-018	Support for x-file		O	
SCR-DS-CONTENT-C-019	Support for x-folder		O	

A.2 Server Data Sync Usage of SyncML Representation

Table 7 – Server Common Use Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-CUE-S-001	Support for Archive element	6.1.1	O	
SCR-DS-CUE-S-002	Support for Chal element	6.1.2	M	SCR-DS-DDE-S-003
SCR-DS-CUE-S-003	Support for Cmd element	6.1.3	M	
SCR-DS-CUE-S-004	Support for CmdId element	6.1.4	M	
SCR-DS-CUE-S-005	Support for CmdRef element	6.1.5	M	
SCR-DS-CUE-S-006	Support for Cred element	6.1.6	M	SCR-DS-DDE-S-001
SCR-DS-CUE-S-007	Support for Field element	6.1.7	O	
SCR-DS-CUE-S-008	Support for Filter element	6.1.8	O	SCR-DS-CUE-S-009 AND (SCR-DS-CUE-S-007 OR SCR-DS-CUE-S-020)
SCR-DS-CUE-S-009	Support for FilterType element	6.1.9	O	
SCR-DS-CUE-S-0010	Support for Final element	6.1.10	M	
SCR-DS-CUE-S-011	Support for Lang element	6.1.11	O	
SCR-DS-CUE-S-012	Support for LocName element	6.1.12	O	
SCR-DS-CUE-S-013	Support for LocURI element	6.1.13	M	
SCR-DS-CUE-S-014	Support for MoreData element	6.1.14	M	

SCR-DS-CUE-S-015	Support for MsgID element	6.1.15	M	
SCR-DS-CUE-S-016	Support for MsgRef element	6.1.16	M	
SCR-DS-CUE-S-017	Support for NoResp element	6.1.17	M	
SCR-DS-CUE-S-018	Support for NoResults element	6.1.18	O	
SCR-DS-CUE-S-019	Support for NumberOfChanges element	6.1.19	M	
SCR-DS-CUE-S-020	Support for Record element	6.1.20	O	
SCR-DS-CUE-S-021	Support for RespURI element	6.1.21	M	
SCR-DS-CUE-S-022	Support for SessionID element	6.1.22	M	
SCR-DS-CUE-S-023	Support for SftDel element	6.1.23	O	
SCR-DS-CUE-S-024	Support for Source element	6.1.24	M	SCR-DS-CUE-S-013
SCR-DS-CUE-S-025	Support for SourceParent	6.1.25	O	SCR-DS-SERVER-012
SCR-DS-CUE-S-026	Support for SourceRef element	6.1.26	M	
SCR-DS-CUE-S-027	Support for Target element	6.1.27	M	SCR-DS-CUE-S-013
SCR-DS-CUE-S-028	Support for TargetParent	6.1.28	O	SCR-DS-SERVER-012
SCR-DS-CUE-S-029	Support for TargetRef element	6.1.29	M	
SCR-DS-CUE-S-030	Support for VerDTD element	6.1.30	M	
SCR-DS-CUE-S-031	Support for VerProto element	6.1.31	M	

Table 8 – Server Message Container Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-MCE-S-001	Support for SyncML element	6.2.1	M	SCR-DS-MCE-S-002 AND SCR-DS-MCE-S-003
SCR-DS-MCE-S-002	Support for SyncHdr element	6.2.2	M	SCR-DS-CUE-S-015 AND SCR-DS-CUE-S-022 AND SCR-DS-CUE-S-024 AND SCR-DS-CUE-S-027 AND SCR-DS-CUE-S-030 AND SCR-DS-CUE-S-031
SCR-DS-MCE-S-003	Support for SyncBody element	6.2.3	M	SCR-DS-PCE-S-001 AND SCR-DS-PCE-S-002 AND SCR-DS-PCE-S-005 AND SCR-DS-PCE-S-

				007 AND SCR-DS-PCE-S-010 AND SCR-DS-PCE-S-011 AND SCR-DS-PCE-S-015 AND SCR- DS-PCE-S-016
--	--	--	--	---

Table 9 – Server Data Description Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-DDE-S-001	Support for Data element	6.3.1	M	
SCR-DS-DDE-S-002	Support for Item element	6.3.2	M	
SCR-DS-DDE-S-003	Support for Meta element	6.3.3	M	SCR-DSDM-METINF-S-011

Table 10 –Server Protocol Management Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-PME-S-001	Support for Status element	6.4.1	M	SCR-DS-CUE-S-003 AND SCR- DS-CUE-S-004 AND SCR-DS- CUE-S-005 AND SCR-DS-CUE- S-015 AND SCR-DS-CUE-S-024 AND SCR-DS-CUE-S-026 AND SCR-DS-DDE-S-001 AND SCR- DS-DDE-S-002

Table 11 – Server Protocol Elements

Item	Functionality	Reference	Status	Requirement
SCR-DS-PCE-S001	Support for Add element	6.5.1	M	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-002	Support for Alert element	6.5.2	M	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-003	Support for Atomic element	6.5.3	O	SCR-DS-CUE-S-004
SCR-DS-PCE-S-004	Support for Copy element	6.5.4	M	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-005	Support for Delete element	6.5.5	M	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-006	Support for Exec element	6.5.6	O	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-007	Support for Get element	6.5.7	M	SCR-DS-CUE-S-004 AND SCR- DS-DDE-S-002
SCR-DS-PCE-S-008	Support for Map element	6.5.8	M	SCR-DS-CUE-S-004 AND SCR-

				DS-PCE-S-009
SCR-DS-PCE-S-009	Support for MapItem element	6.5.9	M	SCR-DS-CUE-S-024 AND SCR-DS-CUE-S-027
SCR-DS-PCE-S-010	Support for Move element	6.5.10	O	
SCR-DS-PCE-S-011	Support for Put element	6.5.11	M	SCR-DS-CUE-S-004 AND SCR-DS-DDE-S-002
SCR-DS-PCE-S-012	Support for Replace element	6.5.12	M	SCR-DS-CUE-S-004 AND SCR-DS-DDE-S-002
SCR-DS-PCE-S-013	Support for Result element	6.5.13	M	SCR-DS-CUE-S-004 AND SCR-DS-DDE-S-002
SCR-DS-PCE-S-014	Support for Search element	6.5.14	O	SCR-DS-CUE-S-004 AND SCR-DS-CUE-S-024 AND SCR-DS-DDE-S-002 AND SCR-DS-DDE-S-003
SCR-DS-PCE-S-015	Support for Sequence element	6.5.15	M	SCR-DS-CUE-S-004
SCR-DS-PCE-S-016	Support for Sync element	6.5.15	M	SCR-DS-CUE-S-004

Table 12 – Server Content Formats

Item	Functionality	Reference	Status	Requirement
SCR-DS-CONTENT-S-001	Support for Contact Synchronization	ALL*	O	SCR-DS-CONTENT-S-008 AND SCR-DS-CONTENT-S-009
SCR-DS-CONTENT-S-002	Support for Calendar Synchronization	ALL*	O	SCR-DS-CONTENT-S-010 AND SCR-DS-CONTENT-S-011
SCR-DS-CONTENT-S-003	Support for Memo Synchronization	ALL*	O	SCR-DS-CONTENT-S-012
SCR-DS-CONTENT-S-004	Support for Task Synchronization	ALL*	O	SCR-DS-CONTENT-S-013
SCR-DS-CONTENT-S-005	Support for Email Synchronization	ALL*	O	SCR-DS-CONTENT-S-014 OR SCR-DS-CONTENT-S-015 OR SCR-DS-CONTENT-S-016 OR SCR-DS-CONTENT-S-017
SCR-DS-CONTENT-S-006	Support for File Synchronization	ALL*	O	SCR-DS-CONTENT-S-018
SCR-DS-CONTENT-S-007	Support for Folder Synchronization	ALL*	O	SCR-DS-CONTENT-S019
SCR-DS-CONTENT-S-008	Support for vCard 2.1	ALL*	O	
SCR-DS-CONTENT-S-009	Support for vCard 3.0	ALL*	O	

SCR-DS-CONTENT-S-010	Support for vCalendar 1.0	ALL*	O	
SCR-DS-CONTENT-S-011	Support for iCalendar 2.0	ALL*	O	
SCR-DS-CONTENT-S-012	Support for text/plain	ALL*	O	
SCR-DS-CONTENT-S-013	Support for vTodo 1.0	ALL*	O	
SCR-DS-CONTENT-S-014	Support for message/rfc822	ALL*	O	
SCR-DS-CONTENT-S-015	Support for message/rfc2822	ALL*	O	
SCR-DS-CONTENT-S-016	Support for Message/rfc2045	ALL*	O	
SCR-DS-CONTENT-S-017	Support for x-email	ALL*	O	
SCR-DS-CONTENT-S-018	Support for x-file	ALL*	O	
SCR-DS-CONTENT-S-019	Support for x-folder	ALL*	O	

* There are examples which use the various content types throughout the specifications.

Appendix B. Change History

(Informative)

B.1 Approved Version History

Reference	Date	Description
OMA-SyncML-DataSyncRep-V1_1_2-20030612-A	12 June 2003	Initial OMA release

B.2 Draft/Candidate Version version 1.2 History

Document Identifier	Date	Sections	Description	
Draft Versions OMA-SyncML-DataSyncRep-V1_2	15 Dec 2003		Initial changes for version 1.2 – Hierarchical, filtering, folders, update version number	
	26 Dec 2003		Updates from review comments. Update to new template, addition of SCR section. Section on Base Media Types from OMA-SyncML-RepPro added.	
	14 Jan 2004		Updates from review comments; Added “Requirement” column information to SCR tables.	
	19 Jan 2004		Changed dates, title and fixed references.	
	03 mar 2004		Clerical changes from consistency review	
	23 apr 2004	6.5.16, 5.12, 5.13, 6.1.19, 8, 6.5.13, 6.5.11, 6.1.3, 6.1.14, 6.3.1, 6.3.3,		CR 62 Sync element Conformance change; CR 58 FilterType default behaviour fix; CR 53 NumberOfChanges conformance error plus additional change in 6.1.19 from “should” to MUST; CR 35 Update DataSyncRep; CR 64 DS SyncHDR Conformance Fix;
	24 apr 2004	5.12, 6.1, 6.5		CR 42 SourceParent/TargetParent Clarification
	25, 26 apr 2004	various		Re-reviewed
	30 apr	Title Page, 1		Updated date; Updated Scope to address Consistency Review #3(1, 3); Re-Reviewed and Updated the usage of several keywords from RFC2119, including MUST, etc.
Candidate Version OMA-SyncML-DataSyncRep-V1_2	01 Jun 2004	n/a	Status changed to Candidate by TP TP ref # OMA-TP-2004-0178-DS-V1_2-for-candidate	