SyncML Server Alerted Notification

Approved Version 1.2.1 – 13 Aug 2007

Open Mobile Alliance

OMA-TS-SyncML_SAN-V1_2_1-20070813-A

Contents

.

Figures

.

1. Scope

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

SyncML is a specification that contains the following main components:

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

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

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

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

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

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

This document describes how a SyncML server may notify a client to initiate a SyncML session.

2. References

2.1 Normative References

[DMBOOT] "SyncML Device Management Bootstrap, Version 1.1.2",Open Mobile Alliance™ URL:http://www.openmobilealliance.org/

[DMNOTI] "Notification Initiated Session, Version 1.1.2", Open Mobile Alliance™. OMA-SyncML-DMNotification-V1_1_2. URL:http:www.openmobilealliance.org/

[DMSEC] "SyncML Device Management Security, Version 1.1.2", Open Mobile Alliance™ URL:http://www.openmobilealliance.org/

[DSPROTO] "DS Protocol, Version 1.2",, Open Mobile Alliance™. OMA-TS-DS_Protocol-V1_2. URL:http:www.openmobilealliance.org/

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

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

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

2.2 Informative References

[RFC1321] "The MD5 Message-Digest Algorithm". Network Working Group. April 1992. http://www.ietf.org/rfc/rfc1321.txt

[RFC1521] "MIME (Multipurpose Internet Mail Extensions) Part One". Network Working Group. September 1993. http://www.ietf.org/rfc/rfc1521.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998, URL:http://www.ietf.org/rfc/rfc2396.txt

[WAPARCH] "WAP Architecture". Open Mobile Alliance™. WAP-210-WAPArch. URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

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

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

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

3.2 Definitions

Application	A SyncML application that supports the SyncML protocol. The application can either be the originator or recipient of the SyncML protocol commands. The application can act as a SyncML client or a SyncML server.
Capabilities exchange	The SyncML capability that allows a client and server to exchange what device, user and application features they each support.
Client	A SyncML Client refers to the protocol role when the application issues SyncML "request" messages. For example in data synchronization, the Sync command in a SyncML Message.
Command	A SyncML Command is a protocol primitive. Each SyncML Command specifies to a recipient an individual operation that is to be performed. For example, the SyncML Commands supported by this specification include Add, Alert, Atomic, Copy, Delete, Exec, Get, Map, Replace, Search, Sequence and Sync.
Data	A unit of information exchange, encoded for transmission over a network.
Data collection	A data element which acts as a container of other data elements, (e.g., {c {{i1, data1}, ... {in, datan}}}). In SyncML, data collections are synchronized with each other. See data element.
data element	A piece of data and an associated identifier for the data, (e.g., {i, data}).
Data element equivalence	When two data elements are synchronized. The exact semantics is defined by a given data synchronization model.
Data exchange	The act of sending, requesting or receiving a set of data elements.
Data format	The encoding used to format a data type. For example, characters or integers or character encoded binary data.
Data type	The schema used to represent a data object (e.g., text/calendar MIME content type for an iCalendar representation of calendar information or text/directory MIME content type for a vCard representation of contact information).
Data synchronization	The act of establishing an equivalence between two data collections, where each data element in one item maps to a data item in the other, and their data is equivalent.
Data synchronization protocol	The well-defined specification of the "handshaking" or workflow needed to accomplish synchronization of data elements on an originator and recipient data collection. The SyncML specification forms the basis for specifying an open data synchronization protocol.
Message	A SyncML Message is the primary contents of a SyncML Package. It contains the SyncML Commands, as well as the related data and meta-information. The SyncML Message is an XML document.
Notification Initiated Session	Device Management terminology for Server Alerted Notification.

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 needed to support SyncML. However, a SyncML implementation that integrates an XML parser might be easier to enhance. This document assumes that the reader has some familiarity with XML syntax and terminology.
Recipient	The network device that receives a SyncML request, processes the request and sends any resultant SyncML response.
Representation protocol	A well-defined format for exchanging a particular form of information. SyncML is a representation protocol for conveying data synchronization and device management operations.
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.
Server Alerted Notification	The method by which a SyncML server notifies a SyncML client to initiate a SyncML session
Server Alerted Sync	Data Synchronization terminology for Server Alerted Notification.
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.
Usage	Refers to the specific usage parts of the SyncML protocol, i.e. Data Synchronization or Device Management

3.3 Abbreviations

WAP	Wireless Application Protocol
ABNF	Augmented Backus-Naur Form [RFC2234]
MD5	A Message Digest algorithm
MIME	Multi-purpose Internet Mail Extensions

4. Introduction

Many devices cannot continuously listen for connections from a server. Other devices simply do not wish to "open a port" (i.e. accept connections) for security reasons. However, most devices can receive unsolicited packages, sometimes called "notifications". Some handsets, for example, can receive SMS packages. Other devices may have the ability to receive other similar datagram packages.

This specification describes the Server Alerted Notification package and associated behaviour. This package is intended to provide the means for a server to notify a client to start a SyncML session with the server. A typical example would be a SyncML Device Management server sending a notification to a SyncML Device Management client device informing it to start a Device Management session with that server.

Note that the general term for this mechanism is *Server Alerted Notification*; for Data Synchronzation the term *Server Alerted Sync* is used, whereas in Device Management the term used is *Notification Initiated Session*.

5. Server Alerted Notification Session

The notification package is intended to provide the means for a server device to notify a client device to start a SyncML session. When the server notifies the client it can tell, for example, the protocol version and whether the server proposes the session to be a foreground or background event.

Figure 1 depicts how a server can initiate a session by sending a Notification Package.

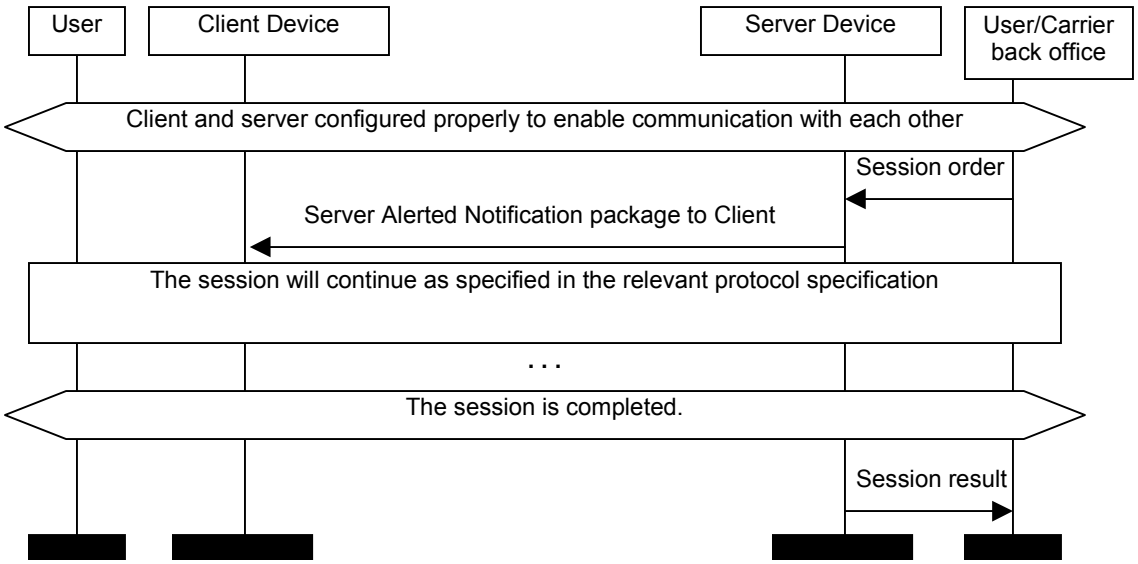

Figure 1. MSC of the Server Alerted Notification session

6. Structure of the Server Alerted Notification Package

Figure 2 describes the format of the Server Alerted Notification Package.

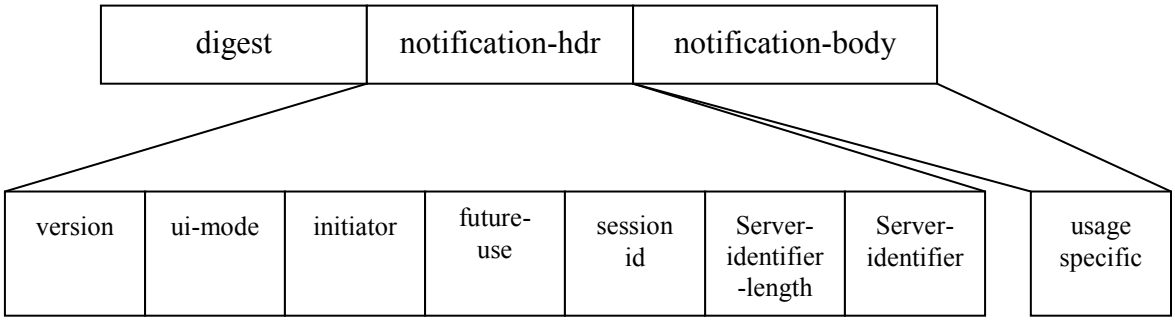

Figure 2. Format of the Server Alerted Notification Package (Package#0)

A Notification Package comprises three parts: a notification header, notification body and a digest of the whole notification. The format of the digest and notification header is the same in both SyncML Device Management and SyncML Data Synchronization usage but the format of the notification body is dependent upon the actual usage-specific protocol. The notification body format for Device Management is described in **[DMNOTI]** and for Data Synchronization in **[DSPROTO]**.

The protocol-specific usage of the Notification Package is indicated by the MIME content type, which is *application/vnd.syncml.notification* for Device Management and *application/vnd.syncml.ds.notification* for Data Synchronization. The content type token MUST be used instead of the textual representation of the MIME code, that is 0x44 for Device Management and 0x4E for Data Synchronization.

7. Syntax for the Package

The following ABNF [RFC2234] defines the syntax for the package. The order and the size of the fields MUST be as specified in the following syntax of the Notification Package.

<notification-package> ::= <digest><notification>

*<digest> ::= 128*BIT* ; 'MD5 Digest value'

<notification> ::= <notification-hdr><notification-body>

<notification-hdr> ::= <version><ui-mode><initiator><future-use>
<sessionid><server-identifier-length><server-identifier>

*<version> ::= 10*BIT* ; 'Protocol Version'
<ui-mode> ::= <not-specified> / <background> / ; 'Background/Informative
<informative> / <user-interaction> ; User Interaction session'
<not-specified> ::= "00" ; '2*bit value "0"'
<background> ::= "01" ; '2*bit value "1"'
<informative> ::= "10" ; '2*bit value "2"
<user-interaction> ::= "11" ; '2*bit value "3"'
<initiator> ::= <user> / <server> ; 'Server/User initiated'
<user> ::= "0" ; '1*bit value "0"'
<server> ::= "1" ; '1*bit value "1"'
*<future-use> ::= 27*BIT* ; 'Reserved for future use'
*<sessionid> ::= 16*BIT* ; 'Session identifier'
*<server-identifier-length> ::= 8*BIT* ; 'Server Identifier length'
*<server-identifier> ::= <server-identifier-length >*CHAR* ; 'Server Identifier'

<notification-body> ::= [usage specific] ; Definition is usage-specific

7.1 Description of the fields

7.1.1 Notification Package

The <*notification-package*> specifies the content of the package the server sends to the client indicating the server's intent for the client to start a session of the indicated type.

7.1.2 Digest

The <*digest*> field specifies the MD5 Digest authentication.

Let H = MD5 hashing function.

Let Digest = output of the MD5 hashing function.

Let B64 = base64 encoding function.

Digest = H(B64(H(server-identifier:password)):nonce:B64(H(notification)))

Where server-identifier is a server identifier for the server account, password is the password of the account the user holds with the server and nonce is a nonce previously installed onto the device via the appropriate mechanism for the usage type, for example, DM bootstrap, [DMBOOT].

The length of the MD5 Digest is 128 bits. The server MUST send the digest to prevent any Denial of Service (DoS) by replay attacks.

Full details of the form and usage of the digest can be found in [DSPROTO] and [DMSEC].

7.1.3 Notification

The <*notification*> field is the container for the notification-hdr and notification-body fields.

7.1.4 Header of the Notification Package

The <*notification-hdr*> field specifies the header of the Notification Package. The structure of this field is identical regardless of the usage.

7.1.5 Body of the Notification Package

The <*notification-body*> field specifies the body of the Notification Package. The structure of this field is different depending on the usage; for example its contents are different for OMA Device Management and OMA Data Synchronization usage. For details of its contents consult the relevant specific protocol specification.

7.1.6 Version Information

The <*version*> field specifies the version of the notification package and the protocol used by the server. This value is specified by using the 10 bits in the Notification Package. The supported version is counted as Supported Version = DIGIT(*version*)/10, i.e. the bit value is transferred to the numeric and then is divided by ten. Therefore the highest version possible is '102.3' and the version '1.0' is specified as '0000001010'.

7.1.7 User Interaction Mode

The <*ui-mode*> field specifies the server recommendations as to whether the server wants the session to be executed in the background or show an indicator to the user.

The values the User Interaction mode can have are:

- Not specified - The <*not-specified*> value in the <*user-interaction*> field specifies that the server doesn't have a recommendation for this element. This value is specified by using the 2 bits with a value of "00".
- Background session action - The <*background*> value specifies that the server recommends the session actions SHOULD be done as a background event. This value is specified by using the 2 bits with a value of "01".
- Informative session action - The <*informative*> value specifies that the server recommends that the client device announces the beginning of the session to the device user, perhaps by means of a visual or audible alert for example. This value is specified by using the 2 bits with a value of "10".
- User Interaction before the session action - The <*user-interaction*> value specifies that the server recommends that the client prompt the device user for acceptance of the offered session before the session takes place. This value is specified by using the 2 bits with a value of "11".

7.1.8 Initiator of the Notification

The <*initiator*> field specifies how the server has interpreted the initiation of the sending of the package, either because the end user requested it or because the server has actions to perform. A client SHOULD make use of this information where applicable – for example when making a connection back to the server a different billing scheme might be applied depending on whether the user initiated the sending of the notification package or the server did.

The values the Initiator of the action can have are:

- User Initiated action - The <*user*> value specifies that the end user caused the session to start. This value is specified by using 1 bit with a value of "0".
- Server Initiated management action - The <*server*> value specifies that the server (operator, enterprise) caused the session to start. This value is specified by using 1 bit with a value of "1".

7.1.9 Future Use

The <*future-use*> field is reserved for possible use in future versions of the specifications. The reserved space is 27 bits long and the bit value for bits not yet in use MUST be "0".

7.1.10 Session Identifier

The <*sessionid*> field specifies the identifier of the session associated with the Notification Package. This value is specified by using the 16 bits in the Notification Package. Whether the session identifier is mandated to have a value or not is dependent upon the usage. If a value is not mandated then the <sessionid> field MUST contain zero (bit value "0000000000000000").

If a server sends more than one Notification Package in an attempt to initiate the same session then it MUST consistently either send the same session identifier in each package or specify a null value in each. If a client receives more than one package with identical contents then it SHOULD ignore all but one of the packages i.e. it SHOULD only initiate one

SyncML session. The client MAY decide for any reason not to respond to the Notification Package, therefore the server SHOULD NOT expect a response to it. If a server has sent a Notification Package to a client and it does not get any response, the server MAY re-send it.

7.1.11 Length of the Server Identifier

The <*server-identifier-length*> field specifies the length of the <*server-identifier*> field in bytes

7.1.12 Server Identifier

The <*server-identifier*> field specifies an identifier of the server from which the notification package is originated.

7.1.13 Notification body

The <*notification-body*> field's contents are usage dependent.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-SyncML_SAN-V1_2-20070221-A	21 Feb 2007	Status changed to Approved by TP: OMA-TP-2007-0005R03-INP_ERP_SyncML_Common_V1_2_for_Final_Approval
OMA-TS-SyncML_SAN-V1_2_1-20070813-A	13 Aug 2007	Updated with agreed class 3 CRs OMA-DS-Comm_1_2-2007-0004 OMA-DS-2005-0212-

Appendix B. Static Conformance Requirements (Normative)

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

B.1 Client Features

B.1.1 Server Alerted Notification Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-C-001	Support for Server Alerted Notification	5	O	DSDM-SAN-C-002 AND DSDM-SAN-C-003 AND DSDM-SAN-C-007 AND DSDM-SAN-C-008 AND DSDM-SAN-C-009 AND DSDM-SAN-C-015

B.1.2 Package #0 Structure Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-C-002	Use of content type token for Notification Package MIME type	6	O	

B.1.3 Package #0 Syntax Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-C-003	Order and Size of fields	7	O	

B.1.4 Description of fields Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-C-004	User Interaction mode	7.1.7	O	
DSDM-SAN-C-005	User Interaction mode values	7.1.7	O	
DSDM-SAN-C-006	Initiator of the Notification	7.1.8	O	
DSDM-SAN-C-007	Future Use	7.1.9	O	
DSDM-SAN-C-008	Session Identifier	7.1.10	O	
DSDM-SAN-C-009	Session Identifier	7.1.10	O	
DSDM-SAN-C-010	Ignoring Packages	7.1.10	O	
DSDM-SAN-C-011	Initiating a single session	7.1.10	O	
DSDM-SAN-C-012	No response from client	7.1.10	O	
DSDM-SAN-C-013	Server Expecting a response	7.1.10	O	
DSDM-SAN-C-014	Resending Packages	7.1.10	O	

B.2 Server Features

B.2.1 Server Alerted Notification Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-S-001	Support for Server Alerted Notification	5	O	DSDM-SAN-S-002 AND

Item	Function	Ref.	Status	Requirement
				DSDM-SAN-S-003 AND DSDM-SAN-S-007 AND DSDM-SAN-S-008 AND DSDM-SAN-S-009 AND DSDM-SAN-S-015

B.2.2 Package #0 Structure Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-S-002	Use of content type token for Notification Package MIME type	6	O	

B.2.3 Package #0 Syntax Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-S-003	Order and Size of fields	7	O	

B.2.4 Description of fields Requirements

Item	Function	Ref.	Status	Requirement
DSDM-SAN-S-004	User Interaction mode	7.1.7	O	
DSDM-SAN-S-005	User Interaction mode values	7.1.7	O	
DSDM-SAN-S-006	Initiator of the Notification	7.1.8	O	
DSDM-SAN-S-007	Future Use	7.1.9	O	
DSDM-SAN-S-008	Session Identifier	7.1.10	O	
DSDM-SAN-S-009	Session Identifier	7.1.10	O	
DSDM-SAN-S-010	Ignoring Packages	7.1.10	O	
DSDM-SAN-S-011	Initiating a single session	7.1.10	O	
DSDM-SAN-S-012	No response from client	7.1.10	O	
DSDM-SAN-S-013	Server Expecting a response	7.1.10	O	
DSDM-SAN-S-014	Resending Packages	7.1.10	O	

B.2.5 Description of fields Requirements

Item	Function	Ref.	Status	Requirement
SCR-DSDM-SAN-S-015	Support for sending of message digest	7.1.2	O	