SyncML Notification Initiated Session

Abstract

This document specifies the SyncML Device Management Notification Initiation package from the server. A management server can use this notification capability to cause the client to initiate a connection back to the management server.

SyncML Initiative

The following companies are Sponsors of the SyncML Initiative:

Ericsson
IBM
Lotus
Matsushita Communication Industrial Co., Ltd.
Motorola
Nokia
Openwave
Starfish Software
Symbian

Revision History

Revision	Date	Comments
V0.8	2001-11-21	Alpha release
V1.1	2002-02-15	First release

Copyright Notice

Table of Contents

1 Introduction

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

A management server can use this notification capability to cause the client to initiate a connection back to the management server. This connection might be over HTTP, WAP or another transport protocol.

The contents of such a “Notification Initiation Alert” might be empty, but the message itself may be signed such that the client can authenticate it. The result of receiving such an alert would be for the client to initiate a connection to the management server that sent the alert. In this scenario, the client might verify that this management server is among those authorized to request such activity. Alternatively, the contents of the alert might indicate that another management server should be contacted.

An identical effect of receiving a Notification Initiation Alert can also be caused in other ways. For example, the user interface (UI) of the device may allow the user to tell the client to initiate a management session. Or, the management client might initiate a session as the result of a timer expiring. Of course, a fault of some type in the device could also cause the management client to initiate a session.

2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interoperated as described in [1].

2.1 Message Sequence Chart Notation

Notation used in the message sequence charts (MSC):

Box – Indicates the start of a procedure or an internal process in a device

Hexagon – Indicates a condition that is needed to start the transaction below this hexagon

Arrow – Represents a message, or transaction

3 Server Alerted Management session

This notification message is intended to provide a possibility for the server to alert the client to perform a management session. When the server alerts the client, it can tell for example the protocol version, whether the server proposes the session to be a foreground or background event. It can also tell if the session is happening because server has some management objects to update or if the user caused the start of the session. The server MUST also send a digest that is included to prevent the Denial of Service (DoS) attacks.

Figure 1 describes the MSC how the server alerts management session.

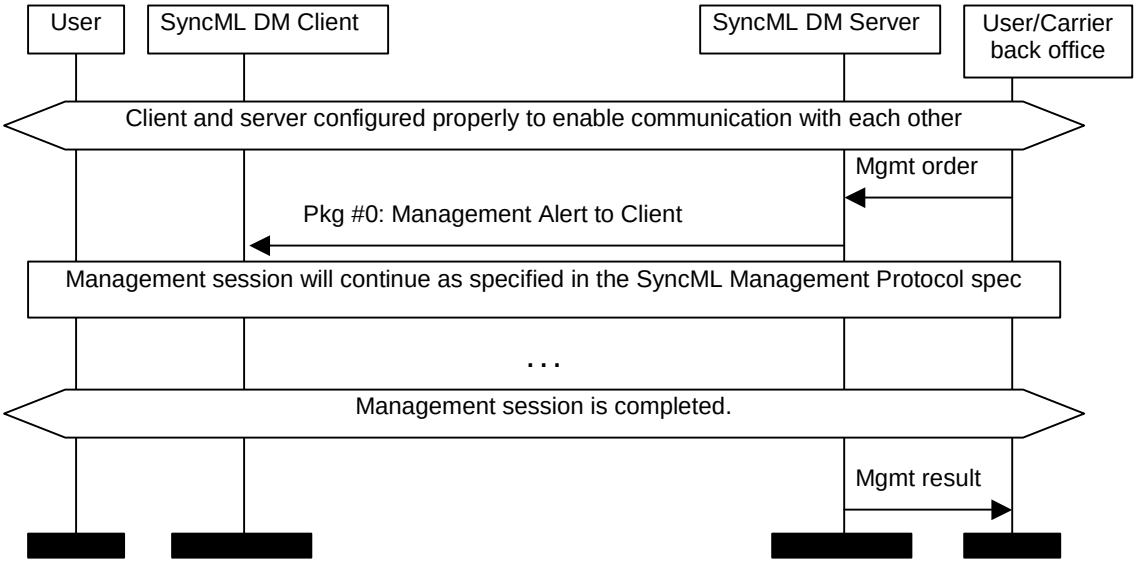

Figure 1. MSC of the Server Alerted Management session

The package flow presented above is one SyncML Device Management session that means that all messages have the same SyncML Session ID.

4 Structure of General Notification Initiated Session Alert

General Package#0 is the default format used for the Notification Initiated Session Trigger Message. This default format can be used if this document does not describe a special format for initialisation purposes.

The following figure describes the format of the General Package #0.

Figure 2. Format of the General Notification Trigger Message (Package#0)

The MIME type for the General Notification Initiated Session Alert message is *application/vnd.syncml.notification* and the Content-Type code for that is *0x42.*

4.1.1 Syntax for the Initiation Notification

The following ABNF [2] defines the syntax for the message. The order and the size of the fields MUST be same as specified in the following syntax of the Trigger Message.

<trigger-message> ::= <digest><trigger>

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

<trigger> ::= <trigger-hdr><trigger-body>

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

*<version> ::= 10*BIT*	; 'Device Management 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> ::= <client> / <server>	; 'Server/User initiatied'
<client> ::= "0"	; '1*bit value "0"'
<server> ::= "1"	; '1*bit value "1"'
*<future-use> ::= 27*BIT*	; 'Reserved for future DM use'
*<sessionid> ::= 16*BIT*	; 'Session identifier'
*<length-identifier> ::= 8*BIT*	; 'Server Identifier length'

<trigger-body> ::= <server-identifier>[<vendor-specific>]

*<server-identifier> ::= <length-source>*CHAR*	; 'Server Identifier'
*<vendor-specific> ::= n*BIT*	; 'Optional vendor specific info'

4.1.2 Description of the fields

4.1.2.1 Trigger Message

The *<trigger-message>* field specifies the message causing the client to connect to the server.

4.1.2.2 Digest

The *<digest>* field specifies the MD5 Digest authentication. The Digest is computed as Digest = H(B64(H(server-identifier:password)):nonce:B64(H(trigger))). Length of MD5 Digest is 128 bits.

4.1.2.3 Trigger

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

4.1.2.4 Header of the Trigger Message

The *<trigger-hdr>* field specifies the header of the Trigger Message.

4.1.2.5 Body of the Trigger Message

The *<trigger-body>* field specifies the body of the Trigger Message.

4.1.2.6 Version information

The <*version*> field specifies the version of the SyncML Device Management Package #0 and the SyncML DM protocol used in the SyncML DM server. This value is specified by using the 10 bits in the Trigger Message. The supported version is counted as Supported version = DEC (*version*)/10. So, first the bit value is transferred to the numeric and then divided by ten. So basically the biggest possible version is '102.3' and the version '1.0' is specified as '0000001010'.

The backward compatibility requirements for protocol versions are specified in more detail in the SyncML Interoperability Testing Process [3].

4.1.2.7 User Interaction mode

The <*ui-mode*> field specifies the server recommendations whether the server wants the management session to be executed in background or show a notification to the user. A client SHOULD follow this recommendation.

The values the User Interaction mode can have:

- Not specified - The <*not-specified*> field in <*user-interaction*> field specifies that the server don't have a recommendation to this element. This value is specified by using the 2 bits and the bit value for not specified action is "00".

- Background management action - The <*background*> field specifies that the server recommends the management action SHOULD be done as a background event. This value is specified by using the 2 bits and the bit value for background action is "01".

- Informative management action - The <*informative*> field specifies that the server recommends the client to display an informative notification or maybe emitting a beep sound announcing the beginning of the provisioning session to the device user. This value is specified by using the 2 bits and the bit value for informative notification is "10".

- User Interaction before the management action - The <*user-interaction*> field specifies that the server recommend the client to prompt the device user for acceptance of the offered management session before the management session takes place. This value is specified by using the 2 bits and the bit value for user displayable notification is "11".

4.1.2.8 Initiator of the Management action

The <*initiator*> field specifies the server recommendations whether the management action is initiated because the end user requested it or because the server has management objects to manage. A client SHOULD follow this recommendation.

The values the Initiator of the Management action can have:

- Client (End User) Initiated management action - The <*client*> field specifies that the end user caused the device management session to start. This value is specified by using 1 bit and the bit value for end user initiated management session is "0".

- Server Initiated management action - The <*server*> field specifies that the server (operator, enterprise) caused the device management session to start. This value is specified by using 1 bit and the bit value for Server initiated management session is "1".

The <client> and <server> values do not convey any information related to "sync type" (SyncML Synchronization protocol document [4] for details of Sync Types).

4.1.2.9 Future use of the Device Management

The <*future-use*> field is reserved for the future fields for SyncML Device Management. The reserved space is 27 bits long and the bit value for bits not yet in use MUST be “0”.

4.1.2.10 Session identifier

The <*sessionid*> field specifies the identifier of the SyncML DM session associated with the SyncML Message. This value is specified by using the 16 bits in the Trigger Message. The Session ID MUST be different between different management session Trigger Messages and the Client MUST use this Session ID when it connects to the SyncML DM Server. If the server triggers the same management session several times, it is recommended the same Session ID be used.

4.1.2.11 Length of the Identifier

The <*length-identifier*> field specifies the length of the Server Identifier of the management server. The value of the Length Identifier is counted as Length of the *server-identifier* = DEC (*length-identifier*).

4.1.2.12 Server Identifier

The <*server-identifier*> field specifies the Server Identifier of the management server. Length of source is specifies in the <*length-source*> field.

4.1.2.13 Vendor specific information

The optional <*vendor specific*> field is used to specify vendor specific information. This field is after the source field and the rest of the Trigger Message size can be fulfilled by the vendor specific information.

5 SyncML Device Management Transport Dependant Profiles

The following sections illustrate the transport dependant profiles for sending a trigger from SyncML Device Management Server to a SyncML Device Management Client.

5.1 Package #0 delivered using WAP Push

The WAP Push framework provides a means for a *Push Initiator* (PI) to send information to a mobile terminal via a *Push Proxy Gateway* (PPG) in an asynchronous manner (see [5] for an overview). It is assumed that the SyncML DM server will act as a PI, but it is also possible for the server to communicate directly with the mobile terminal if it is able to operate as a PPG.

When the WAP Push framework is used to deliver Package #0, the non-secure connectionless WSP [8] session service is utilised as defined in [6]. The following rules MUST be adhered to as well as the order of the WSP headers:

- The Content-Type header [7] MUST include the MIME media type for Packet #0 as defined in [9]. The Content-Type code 0x42 registered with WINA [10] MUST be used instead of the textual representation of the MIME code.
- The X-WAP-Application-ID header [7] MUST include the application-id associated with the Sync ML Device Management User Agent (registered with WINA [10]). The application-id code 0x07 MUST be used instead of the textual representation of the Application-id.
- Other headers may be included if it is known that the SyncML DM Client can interpret them in a useful manner. However, it must be ensured that the total length of the WDP and WSP headers never exceeds 48 bytes to ensure that there is sufficient space for the payload.
- The push message is sent to the default non-secure connectionless push port (2948)

The message payload has been designed to fit into a single short message when SMS is used to deliver WAP Push. If the WAP Push message does not fit into a single SMS message the concatenated messages MUST be used.

5.1.1 Using non WAP Push capable devices

If the receiver is not a WAP device, it is very unlikely that any other application would be active on the same port, which has been publicly registered with IANA. The decoding of the message headers is very straightforward even if the device lacks a full WAP stack and therefore the device MUST examine if the message has been sent to the WAP push port (2948) and if the Application-ID and the MIME type are one assigned to the SyncML DM Notification Initiation Package. If this information is correct then the message MUST be routed to the SyncML Device Management application.

5.2 Package #0 over OBEX

Local Notification Initiated Session over OBEX is done inside the PUT command of the OBEX protocol. This happens in the same way as sending the SyncML messages over OBEX to a SyncML client (See the SyncML OBEX Binding specification [11]).

6 References

[1] Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, IETF.

[2] Augmented BNF for Syntax Specifications, RFC 2234, IETF.

[3] SyncML Interoperability Testing Process, SyncML.

[4] SyncML Synchronization Protocol Specification, Version 1.1, SyncML.

[5] WAP Push Architectural Overview, WAP Forum.

[6] Push OTA Protocol, WAP Forum.

[7] Push Message, WAP Forum.

[8] WAP Wireless Session Protocol Specification, WAP Forum.

[9] Internet Assigned Numbers Authority, IANA.

[10] WAP Interim Naming Authority, WINA.

[11] SyncML OBEX Binding Specification, Version 1.1, SyncML.

7 Appendix 1 - Example of Trigger Message from Server

Example WAP Push over SMS containing the trigger information:

Binary value	Meaning	Description
06	User-Data-Header (UDHL) Length = 6 bytes	WDP layer (start WDP headers).
05	UDH IE identifier: Port numbers	
04	UDH port number IE length	
0B	Destination port (high)	Port number 2948
84	Destinating port (low)	
C0	Originating port (high)	Port number chosen by sender
02	Originating port (low)	WDP layer (end WDP headers)
01	Transaction ID / Push ID	WSP layer (start WSP headers)
06	PDU type (push)	
03	Headerslength (content type+headers)	
C2	Content type code	MIME-Type
AF	X-WAP-Application-ID	
87	Id for urn: x-wap-application:syncml.dm	WSP layer (end WSP headers)
	128-bit digest value	Digest
	Binary ‘0000001011’	Version ‘1.1’
	Binary ‘01’	UI-Mode ‘1’
	Binary ‘00’	Initiator ‘0’
	Binary ‘00000000000000000000000000’	Future DM use
	Binary ‘0000000000000001’	SessionID ‘1’
	Binary ‘0000100010’	Server ID length ‘34’
68, 74, 74, 70, 3A, 2F, 2F, 77, 77, 77, 2E, 6D, 6E, 67, 6D, 74, 73, 65, 72, 76, 65, 72, 2E, 63, 6F, 6D, 2F, 6D, 61, 6E, 61, 67, 65, 2F	String ‘http://www.mngmtserver.com/manage/’	Server Identifier