OMA DS Notification

Approved Version 2.0 – 19 Jul 2011

Open Mobile Alliance

OMA-TS-DS_Notification-V2_0-20110719-A

Contents

Figures

.

1. Scope

This document describes how a DS Server may notify a DS Client to initiate a synchronization session.

Please refer to [DSCONCEPTS] for further information on the OMA DS organization and history.

2. References

2.1 Normative References

[DEVINF] "OMA DS Device Information ", Open Mobile Alliance™, OMA-TS-DS-DevInfo-V2_0", URL:http://www.openmobilealliance.org

[DMBOOT] "OMA Device Management Bootstrap, Version 1.2", Open Mobile Alliance™. OMA-TS-DM_Bootstrap-V1_2. URL:http://www.openmobilealliance.org/

[DMPRO] "OMA Device Management Protocol", Version 1.2, Open Mobile Alliance™, OMA-TS-DM_Protocol-V1_2, URL:http://www.openmobilealliance.org/

[DMSEC] "OMA Device Management Security, Version 1.2", Open Mobile Alliance™. OMA-TS-DM_Security-V1_2, URL:http://www.openmobilealliance.org

[DSCONCEPTS] "Data Synchronization Concepts and Definitions", Version 2.0, Open Mobile Alliance™, OMA-TS-DS_Concepts-V2_0, URL:http:www.openmobilealliance.org

[DSMO] "Data Synchronization Management Object", Open Mobile Alliance™, OMA-TS-DS_MO-V1_0, URL:http:www.openmobilealliance.org

[DSPRO] "Data Synchronization Protocol", Version 2.0, Open Mobile Alliance™. OMA-TS-DS_Protocol-V2_0, URL:http:www.openmobilealliance.org

[DSSYNTAX] "Data Synchronization Syntax", Open Mobile Alliance™, OMA-TS-DS_Syntax-V2_0, URL:http:www.openmobilealliance.org

[IOPPROC] "OMA Interoperability Policy and Process", Open Mobile Alliance™, OMA-ORG-IOP_Process-V1_4, URL:http//www.openmobilealliance.org

[RFC1321] " The MD5 Message-Digest Algorithm ", R. Rivest, et al., April 1992, URL:http://www.ietf.org/rfc/rfc1321.txt

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

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

[WSPCTC] "WSP Content Type Codes" Open Mobile AllianceTM, URL:http://www.openmobilealliance.org/tech/omna/omna-wsp-content-type.htm

2.2 Informative References

None.

Please refer to [DSCONCEPTS] for the other Informative References.

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 Data Synchronization XML Schema or XML snippets is specified in this `typeface`.

3.2 Definitions

Please refer to the [DSCONCEPTS] document.

3.3 Abbreviations

Please refer to the [DSCONCEPTS] document.

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 DS Notification package and associated behaviour. This package is intended to provide the means for a DS Server to notify a DS Client to start a synchronization session with the server. A typical example would be a DS Server sending a notification to a DS Client informing the DS Client to start a synchronization session with that server.

4.1 Version 2.0

This is the first version of the Notification specification. The reason for the specification version number 2.0 is to be consistent with the service release version number, that is, DS 2.0.

The Notification specification is derived from the Server Alerted Notification mechasnism defined in DS 1.2.1. Based on that, it makes the following enhancements: nonce resynchronization, more action types, more extendable, notification acknowledgement.

The Notification mechanism is backward compatible with the Server Alerted Notification mechanism defined in DS 1.2.1.

5. DS Notification Initiated Session

The notification package is intended to provide the means for a DS Server to notify a DS Client to start a sycnhronization 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 DS Server can initiate a session by sending a Notification Package.

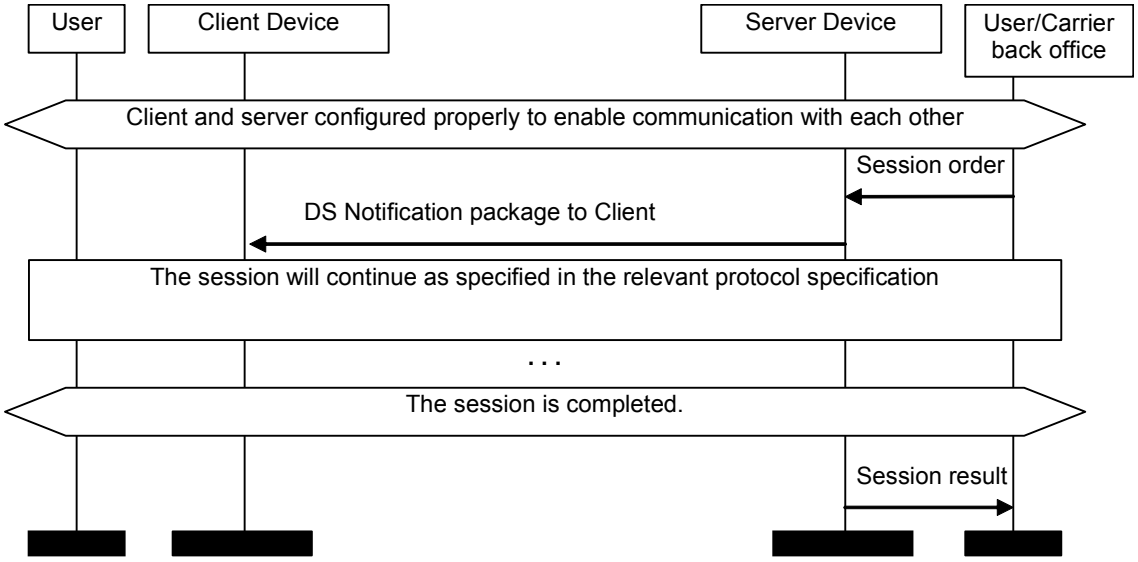

Figure 1. Message Sequence Chart of the DS Notification session

5.1 Nonce Resynchronisation

After the client has received a notification message and the digest is not correct then the client SHOULD re-verify the digest using the <*SessionID*> as the nonce value. Servers can not expect all clients to supports this features and MAY therefore take a different action, such as re-bootstrap the device, for example.

The flow of this particular scenario is as follows:

1. Client receives Notification message and fails to authenticate the message using stored server nonce value.

2. Client tries to re-authenticate notification message using a default server nonce value (<*SessionID*>). If this authentication succeeds, then continue to step 3; otherwise notification message is ignored and no session is initiated.

3. Client initiates session, with the credentials based on client nonce provided by Server in advance in case of application layer security.

4. Server tries to authenticate the message. If this authentication succeeds, server sends a success `Status` code with a `Chal` to update the client nonce on the device used to authenticate messages from the client and then continue to step 5. If this authentication fails, then Server sends back `Chal` to request Client to re-send credential for authentication and then continue to step 3.

5. Client tries to authenticate the message, in case of application layer security then with the default nonce value (<*SessionID*>). If this authentication succeeds, then Client sends success Status code with a `Chal` to update the server nonce on the server used by the client to authenticate messages from the server (including notification messages).

6. Server sends back success `Status`.

If desired, server replaces the server `AuthSecret` in the DS MO (see [DSMO]) using the DM protocol (see [DMPRO]) to further protect against repeated attacks on the nonce re-negotiation protocol.

6. DS Notification Security

DS Notification Package MUST be authenticated and MD5 Digest authentication MUST be used.

In case the client does not receive the confirmation for the package that delivers the new nonce to the server, the client MUST NOT discard the old nonce as it is unknown on the client side whether the server already received the new nonce or not. Instead, the client MUST keep both nonces until the new sync notification received from the server and try to authenticate the Notification Package using the old nonce if the authentication with the new one fails. If the authentication fails with both nonces, the client MAY try to authenticate the Notification Package using the default nonce (see section 5.1). If the authentication fails with the above three nonces, the Notification Package is assumed from non-authorized sender and nonces MUST be left intact. If the authentication succeeds, the invalid nonce MUST be discarded on the client and the valid nonce MUST be stored as “old” until the new nonce is created and successfully delivered to the server.

The server SHOULD send the Notification Package using the stored server nonce. In case that the server stored nonce mismatches with both the old and the new nonce stored by the client, the server MAY send the Notification Package using the default nonce (see section 5.1). Note that how can the server judge nonce mismatch will depend on server policy, for example, the server does not receive the client initiated session for a long time after sending the Notification Package.

The old nonce MUST be discarded on the client only after the client receives a successful confirmation for the package that contains the new nonce for the server’s future use.

7. Structure of the DS Notification Package

Figure 2 describes the format of the DS Notification Package.

A Notification Package comprises three parts: a digest of the whole notification package, a notification header (see Figure 3), and a notification body (see Figure 4).

The protocol-specific usage of the DS Notification Package is indicated by the MIME content type, which is *application/vnd.syncml.ds.notification*.

The content type token MUST be used instead of the textual representation of the MIME content type. The content type token for OMA DS is 0x4E.

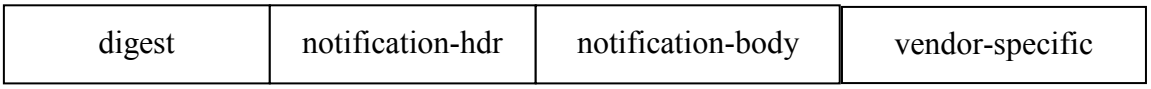

Figure 2. Format of the DS Notification Package

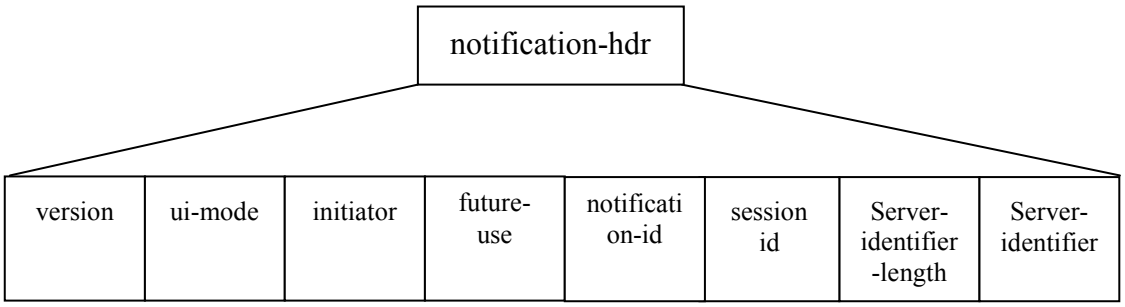

Figure 3. Format of the DS Notification Package Header (<notification-hdr>)

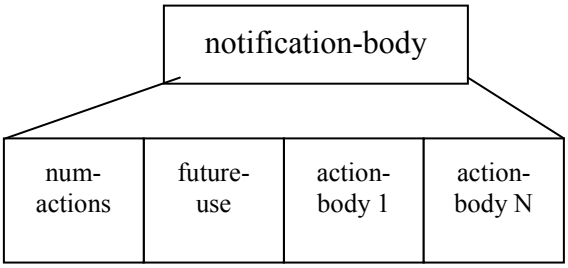

Figure 4. Format of the DS Notification Package Body (<notification-body>)

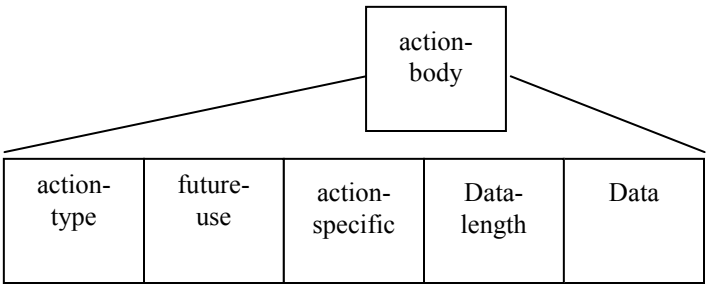

Figure 5. Format of the Action Package Body (<action-body>)

8. Syntax for the Notification 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 DS Notification Package.

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

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

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

<notification-hdr> ::= <version><ui-mode><initiator><future-use><notification-id>
<session-id><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> ::= 11*BIT* ; 'Reserved for future use'
*<notification-id> ::= 16*BIT* ; 'Notification identifier'
*<session-id> ::= 16*BIT* ; 'Session identifier'
*<server-identifier-length> ::= 8*BIT* ; 'Server Identifier length'
*<server-identifier> ::= <server-identifier-length >*CHAR* ; 'Server Identifier'

*<notification-body> ::= <num- actions><future-use><action-body>** *; 'Body Data'*
*<num-actions> ::= 4*BIT* *; 'Number of actions'*
*<future-use> ::= 4*BIT* *; 'Reserved for future use'*

<action-body > ::= <action-type><future-use><action-specific>
<Data-length><Data> *; 'Action Information'*
*<action-type> ::= 4*BIT* *; 'Action type'*
*<future-use> ::= 4*BIT* *; 'Reserved for future use'*
*<action-specific> ::= 24*BIT* *; 'Action specific information'*
*<Data-length> ::= 8*BIT* *; 'Data Length'*
*<Data> ::= <Data-length >*CHAR* *; 'Data'*
s
*<vendor-specific> ::= n*BIT* *; 'Optional vendor-specific information'*

9. Description of the DS Notification fields

9.1 <notification-package>

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

Note that the length of the Notification Package will be limited subject to the transport bearers. For different transport bearers, the limitations for the length of the Notification Package will be different.

9.1.1 <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 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 client 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 [DSPRO] and [DMSEC].

9.1.2 <notification>

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

9.1.2.1 <notification-hdr>

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

9.1.2.1.1 <version>

This field MUST specify the version of the OMA Data Synchronization protocol that the server supports.

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, e.g.: for version '2.0', the content of <*version*> is '0000010100'. Therefore the highest version possible is '102.3'.

9.1.2.1.2 <ui-mode>

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

The values of <*ui-mode*>field are:

- '00' = Not specified. It indicates that the server doesn't have any requirement for this element.
- '01' = Background session action. It specifies that the client is required to do the session as a background event (i.e.: no announcement of the beginning of the session to the user, nor user interaction).
- '10' = Informative session action. It specifies that the client is required to announce the beginning of the session to the user, by means of a visual or audible alert for example.
- '11' = User Interaction before the session action. It specifies that the client is required to prompt the user for acceptance of the offered session before the session takes place..

9.1.2.1.3 <initiator>

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 than 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".

9.1.2.1.4 <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".

9.1.2.1.5 <notification-id>

The <*notification-id*> field is used to identify the Notification Package. The value of this field is specified by using the 16 bits in the Notification Package.

A server MUST specify a value for this field. The server MAY specify zero value (bit value "0000000000000000"), which indicates the client not to respond to the Notification Package. If <*notification-id*> value is not equal to zero value, the client MUST use this value when it responds to the Notification Package.

If a server sends more than one Notification Package in an attempt to initiate the same session, the server SHOULD specify different <*notification-id*> for each Notification Package. If the client wants to respond to the Notification Package, it is enough for a client to send back only one Acknowledgement Package. Note that clients MAY see multiple identical packages with unreliable transport layers.

If a server sends Notification Package for different session, the <*notification-id*> for different Notification Package MUST be different.

9.1.2.1.6 <session-id>

The <*session-id*> field specifies the identifier of the session associated with the DS Notification Package. This value is specified by using the 16 bits in the Notification Package.

A server MUST specify a value for this field. The server can specify zero value (bit value "0000000000000000"), which indicates the client to define the session identifier.

If <*session-id*> value is not equal to zero value, the client MUST use this value for all subsequent packages in the session.

If a server sends more than one Notification Package in an attempt to initiate the same session then it MUST consistently send the same value for the <*sessionid*> in each package.

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 synchronization session.

The client MAY 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.

9.1.2.1.7 <server-identifier-length>

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

9.1.2.1.8 <server-identifier>

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

Its value MUST be unique, it MUST be the same as the Server Identifier (DevID) in the [DEVINF], it SHOULD contain the server domain name to aid uniqueness, and it SHOULD be kept as short as possible for efficiency and cannot exceed 255 characters in length.

9.1.2.2 <notification-body>

The <*notification-body*> field specifies the body of the Notification Package.

9.1.2.2.1 <num-actions>

The <*num- actions*> field is used to specify the number of <action-body> sections.

If the server is requesting N actions, then the <*notification-body*> MUST contain N <*action-body*> fields.

If <*num-actions*> is zero, it has the following meaning:

- the client SHOULD use a 'twoWay preserve' sync type (value “0000” for <*action-type*>), unless the client can determine a more appropriate action, such as by using stored sync type information.
- the client SHOULD initiate synchronization with all “preknown” data stores of the server. Data stores could be “preknown” as a result of earlier sync session, or they are configured e.g. by an end-user.

9.1.2.2.2 <future-use>

The <*future-use*> field is reserved for future use for OMA Data Synchronization. The reserved space is 4 bits long and the value for bits not yet in use MUST be “0”.

9.1.2.2.3 <action-body>

The <*action-body*> field is a container for the <*action-type*>, <*future-use*>, <*action-specific*>, <*Data-length*> and <*Data*> fields. The number of <*action-body*> fields in the Notification Package MUST be equal to the value specified in the <*num-actions*> field.

9.1.2.2.3.1. <action-type>

The <*action-type*> field specifies the *action* code indicating which action type the server is requesting the client start.

The action types can be data synchronization, device information negotiation, empty session initiation etc.

The values for different action types are listed as the following:

Values	Description
0000~0111	Data synchronization
1000	Device information negotiation
1001	Empty session initiation
1010	Session Information Indication
1011~1111	Reserved for future use

The values from '0000' to '0111' are used for data synchronization purpose.

9.1.2.2.3.2. <future-use>

The <*future-use*> field is reserved for future use for OMA Data Synchronization. The reserved space is 4 bits long and the value for bits not yet in use MUST be "0".

9.1.2.2.3.3. <action-specific>

The <*action-specific*> field specifies the action specific information. For different action types, the meanings of this field are different.

9.1.2.2.3.4. <Data-length>

The <*Data-length*> field specifies the length of the Data in bytes. If the <*Data*> field is not existed, the vaule of <*Data-length* > will be zero.

9.1.2.2.3.5. <Data>

The <*Data*> field specifies the action specific data, which can be data store URI or device information URI depending on the corresponding action types.

The length of this field is specified in the <*Data-length*> field.

Note that the length of the <*Data*> field will be limited subject to the transport bearers. For different transport bearers, the limitations for the length will be different.

9.1.2.3 <vendor specific>

The optional <*vendor specific*> field is used to specify vendor-specific information.

9.2 Action Types

The action types can be data synchronization, device information negotiation, empty session initiation etc.

For different action types, the meaning for the <*future-use*>, <*Data-length*> fields contained in <*action-body*> are the same, but the meanings for the <*action-type*>, <*action-specific*>, <*Data*> fields are different.This section introduces the different meaning of the fields for the different action types.

9.2.1 Data Synchronization

The server can use notification message to indicate the client to initiate a data synchronization session to exchange data items.

9.2.1.1 <action-type>

For the data synchronization action, the value of <*action-type*> field ranged from '0000' to '0111' specifies the different sync types.

The following ABNF [RFC2234] defines the syntax for the <*action-type*> in data synchronization usage.

The order and the size of the fields MUST be as specified in the following syntax:

```
<action-type> ::= <data-sync><direction><behaviour>          ; 'Action Type Information,4*BIT'
<data-sync> ::= "0"                                          ; 'Data Sync,1*BIT'
<direction> ::= <twoWay> / <fromClient> / <fromServer> / <NoWay>   ; 'Direction,2*BIT'
<twoWay> ::= "00"                                            ; 'twoWay'
<fromClient> ::= "01"                                        ; 'fromClient'
<fromServer> ::= "10"                                        ; 'fromServer'
<NoWay> ::= "11"                                             ; 'NoWay'
<behaviour> ::= <preserve> / <refresh>                       ; 'Behaviour,1*BIT'
<preserve> ::= "0"                                           ; 'Preserve'
<refresh> ::= "1"                                            ; 'Refresh'
```

Please note that 'twoWay refresh' sync type is not valid, so "0001" MUST NOT be used for <*action-type*>.

If the client starts a synchronization session using the same session identifier indicated in the Notification Package, the client MAY initiate a different sync type, but the client MUST be compliant to the sync type transition rules specified in the [DSPRO]. If the client starts a synchronization session using a different session identifier, the transition rules don't need to be complied.

9.2.1.2 <action-specific>

For the data synchronization action, the <*action-specific*> field specifies the content type information.

This field specifies the MIME media content type of data object that the server instructs the device to synchronize.

This field SHOULD be used to indicate the content type of the data store the server wishes to synchronize. It MUST contain content type information or contain zero (bit value "000000000000000000000000"). If the content type is being specified then it MUST contain the numeric content type code (described in [WSPCTC]) and not its corresponding textural representation.

9.2.1.3 <Data>

For the data synchronization action, the <*Data*> field specifies the name of the server data store. Relative URI's SHOULD be used for efficiency. It MUST be specified for each data synchronization <*action-body*> item present in the <*notification-body*> field.

9.2.2 Device Information Negotiation

The server can use notification message to indicate the client to initiate a device information negotiation session to exchange device information.

9.2.2.1 <action-type>

For the device information negotiation action, the value of <*action-type*> field is '1000'.

9.2.2.2 <action-specific>

For the device information negotiation action, this field is not used currently and reserved for later usage. And it MUST contain zero (bit value "000000000000000000000000").

9.2.2.3 <Data>

For the device information negotiation action, the <*Data*> field specifies the URI of the device information. In this way, the server can indicate the client to send all the device information or part of the device information. For example, for all the device information, the URI can be './devinfo20/DevInf', and for a specific DataStore information, the URI can be './devinfo20/DevInf/DataStore[DisplayName='addressbook']'.

If the <*Data*> field is not specified, the client SHOULD send the the appropriate device information which the client determines to send, for example, the updated device information.

9.2.3 Empty Session Initiation

The server can use notification message to indicate the client to initiate an empty session back to the server. Based on the empty session, the server can get the client device information or perform data synchronization operations.

9.2.3.1 <action-type>

For the empty session initiation action, the value of <*action-type*> field is '1001'.

9.2.3.2 <action-specific>

For the empty session initiation action, this field is not used currently and reserved for later usage. And it MUST contain zero (bit value "000000000000000000000000").

9.2.3.3 <Data>

For the empty session initiation action, the <*Data*> field is not used. And the <*Data-length*> MUST be zero.

9.2.4 Session Information Indication

In the notification message, the server can provide some session indication information to the client. According to the session indication information, the client or the user can determine whether or not to initiate the session with the server.

9.2.4.1 <action-type>

For the session information indication action, the value of <*action-type*> field is '1010'.

9.2.4.2 <action-specific>

For the session information indication action, this field is not used currently and reserved for later usage. And it MUST contain zero (bit value "000000000000000000000000").

9.2.4.3 <Data>

For the session information indication action, the <*Data*> field specifies the readable session indication information. It indicates the session purpose in readable character string, for example, "You have a new mail!".

10.Acknowledgement of DS Notification Package

In some cases, after receiving the Notification package from server, if the client doesn't want to initiate a session with the server, the client MAY send acknowledgement package to the server. Then the server will not resend the Notification package again. This can avoid unnecessary and unsolicited notifications.

If the client wants to initiate a session with the server, the client SHOULD NOT send acknowledgement package to the server.

The client SHOULD NOT send an acknowledgement package to the server when authentication has failed in the received DS Notification package.

The transport layer for the acknowledgement package SHOULD be the same as the Notification package.

10.1 Structure of the Acknowledgement Package

Figure 6 describes the structure of the DS Acknowledgement Package.

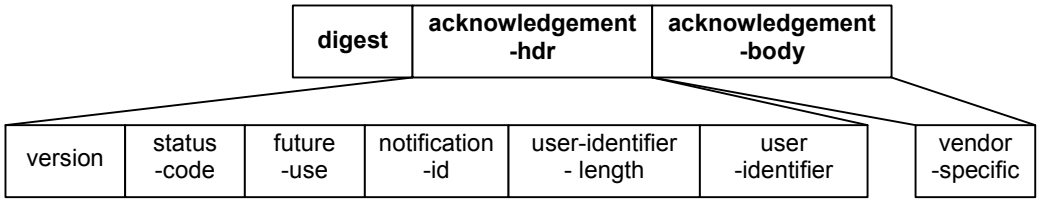

Figure 6 DS Acknowledgement Package

10.2 Syntax of the Acknowledgement Package

The following ABNF [RFC2234] defines the syntax for the acknowledgement package.

The order and the size of the fields MUST be as specified in the following syntax of the DS Acknowledgement Package.

<acknowledgement-package> ::= <digest><acknowledgement>

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

<acknowledgement> ::= <acknowledgement-hdr>< acknowledgement-body>

< acknowledgement-hdr> ::= <version><status-code><future-use>
<notification-id><user-identifier-length><user-identifier>

*<version> ::= 10*BIT* ; 'Protocol Version'
*<status-code> ::= 3*BIT* ; 'Status Code'
*<future-use> ::= 27*BIT* ; 'Reserved for future use'
*<notification-id> ::= 16*BIT* ; 'Notification Identifier'
*<user-identifier> ::= 8*BIT* ; 'User Identifier length'
*<user-identifier-length> ::= <user-identifier>*CHAR* ; 'User Identifier'

<acknowledgement -body> ::= <vendor-specific>

*<vendor-specific> ::= n*BIT* *; 'Vendor-specific information'*

10.3 Description of fields in Acknowledgement Package

The <*acknowledgement-package*> specifies the content of the package that the client sends to the server, indicating the acknowledgement of the notification package.

10.3.1 <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(user-identifier: password)):nonce:B64(H(acknowledgement)))

Where 'user-identifier' is a identifier for the user account, 'password' is the password of the user identifier the server holds with the client and 'nonce' is a nonce previously used in Notification Package.

The length of the MD5 Digest is 128 bits. The client 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 [DSPRO] and [DMSEC].

10.3.2 <acknowledgement-hdr>

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

10.3.2.1 <version>

This field MUST specify the version of the OMA Data Synchronization protocol that the client supports.

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, e.g.: for version ‘2.0’, the content of <*version*> is ‘0000010100’. Therefore the highest version possible is ‘102.3’.

10.3.2.2 <status-code>

The <*status-code*> field specifies the status code. The status code is used to indicate the reason that the client doesn’t initiate a session back to the server.

The values for different status codes are listed as the following:

Values	Description
000	User rejects the session
001	Device is busy
010	Notification format error
011	No specific reason
100~111	Reserved for future use

According to the status code, the server can decide not to resend the Notification package again or reassemble the Notification package according to the specific indication information.

10.3.2.3 <future-use>

The <*future-use*> field is reserved for possible use in future versions of the specifications. The reserved space is 11 bits long and the bit value for bits not yet in use MUST be "0".

10.3.2.4 <notification-id>

The <*notification-id*> field specifies the identifier of the Notification Package. The value of this field MUST be the same as the value of <*notification-id*> field specified in the Notification Package. In this way, the Acknowledgement Package can be linked to the Notification Package.

10.3.2.5 <user-identifier-length>

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

10.3.2.6 <user-identifier>

The <*user-identifier*> field specifies the user identifier which is used in digest computation.

Its value MUST be unique, and it MUST be the same as the User Identifier in the [DSPRO], and it SHOULD be kept as short as possible for efficiency and cannot exceed 255 characters in length.

10.3.3 <acknowledgement-body>

The <*acknowledgement-body*> field specifies the body of the Acknowledgement Package.

10.3.3.1 <vendor specific>

The <*vendor specific*> field is used to specify vendor-specific information.

11.Example of DS Notification Package

<< Data (i.e. notification) pushed from the server >>		
XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX, XX	128-bit digest value	MD5 Digest authentication value
Notification Header >>		
03, 18, 00, 00, 00	Binary ‘0000001100’	Version ‘1.2’
	Binary ‘01’ = Background mode	UI-Mode ‘1’
	Binary ‘1’ = Server Initiated Action	Initiator ‘1’
	Binary ‘00000000000000000000000000’	Future use
00, 01	Binary ‘0000000000000001’	SessionID ‘1’
0F	Binary ‘00001111’	Server Identifier length ‘15’
73, 79, 6E, 63, 2E, 73, 65, 72, 76, 65, 72, 2E, 63, 6F, 6D	String ‘sync.server.com’	Server Identifier
Notification Body >>		
10	Binary ‘0001’	Number of syncs. One Sync Info container included
	Binary ‘0000’	Future use
Sync info >>		
60	Binary ‘0000’	Sync type ‘two way preserve’
	Binary ‘0000’	Future use
00, 00, 07	Binary ‘000000000000000000000111’	Content-type ‘text/x-vcard’
04	Binary ‘00000100’	Server URI length ‘4’
2F, 63, 6F, 6E	String ‘/con’	The name of server data store [Server URI]
*Optional Vendor specific info >> (Binary n*BIT)*		

Appendix A. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [IOPPROC]

A.1 Client Features

A.1.1 DS Notification Requirements

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

A.1.2 DS Notification Structure Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-C-002	Use of content type token for Notification Package MIME type	7	O	

A.1.3 DS Notification Syntax Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-C-003	Order and Size of fields	9	O	

A.1.4 Description of fields Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-C-004	User Interaction mode	9.1.2	O	
DS-SAN-C-005	User Interaction mode values	9.1.2	O	
DS-SAN-C-006	Initiator of the Notification	9.1.2	O	
DS-SAN-C-007	Future Use	9.1.2	O	
DS-SAN-C-008	Session Identifier	9.1.2	O	
DS-SAN-C-009	Server Identifier	9.1.2	O	
DS-SAN-C-010	Ignoring Packages	9	O	
DS-SAN-C-011	Initiating a single session	9	O	
DS-SAN-C-012	No response from client	9	O	
DS-SAN-C-013	Server Expecting a response	9	O	
DS-SAN-C-014	Resending Packages	9	O	
DS-SAN-C-015	Nonce Resynchronziation	5.1	O	
DS-SAN-C-016	Action Types	9.2	O	
DS-SAN-C-017	Acknowledgement of Notification	10	O	

A.2 Server Features

A.2.1 DS Notification Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-S-001	Support for DS Notification	5	O	DS-SAN-S-002 AND DS-SAN-S-003 AND DS-SAN-S-007 AND DS-SAN-S-008 AND DS-SAN-S-009 AND DS-SAN-S-015

A.2.2 DS Notification Structure Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-S-002	Use of content type token for Notification Package MIME type	7	O	

A.2.3 DS Notification Syntax Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-S-003	Order and Size of fields	9	O	

A.2.4 Description of fields Requirements

Item	Function	Ref.	Status	Requirement
DS-SAN-S-004	User Interaction mode	9.1.2	O	
DS-SAN-S-005	User Interaction mode values	9.1.2	O	
DS-SAN-S-006	Initiator of the Notification	9.1.2	O	
DS-SAN-S-007	Future Use	9.1.2	O	
DS-SAN-S-008	Session Identifier	9.1.2	O	
DS-SAN-S-009	Server Identifier	9.1.2	O	
DS-SAN-S-010	Ignoring Packages	9	O	
DS-SAN-S-011	Initiating a single session	9	O	
DS-SAN-S-012	No response from client	9	O	
DS-SAN-S-013	Server Expecting a response	9	O	
DS-SAN-S-014	Resending Packages	9	O	
DS-SAN-S-015	Nonce Resynchronziation	5.1	O	
DS-SAN-S-016	Action Types	9.2	O	
DS-SAN-S-017	Acknowledgement of Notification	10	O	

A.2.5 Description of fields Requirements

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

Appendix B. Change History (Informative)

B.1 Approved Version 2.0 History

Reference	Date	Description
OMA-TS-DS_Notification-V2_0-20110719-A	19 Jul 2011	Status changed to Approved by TP: OMA-TP-2011-0258-INP_DS_V2_0_ERP_for_final_Approval