OMA Device Management Protocol

Candidate Version 1.3 – 09 Oct 2012

Open Mobile Alliance

OMA-TS-DM_Protocol-V1_3-20121009-C

Contents

1. SCOPE 5
2. REFERENCES 6
2.1 NORMATIVE REFERENCES 6
2.2 INFORMATIVE REFERENCES 6
3. TERMINOLOGY AND CONVENTIONS 7
3.1 CONVENTIONS 7
3.2 DEFINITIONS 7
3.3 ABBREVIATIONS 7
4. INTRODUCTION 8
5. PROTOCOL OVERVIEW (INFORMATIVE) 9
5.1 TRANSACTION MODEL 9
5.2 DM BOOTSTRAPPING 9
5.3 MESSAGE ENCODING FORMAT 10
5.4 DEVICE MANAGEMENT TREE 10
5.5 PROTOCOL COMMANDS 10
5.6 DEVICE DESCRIPTION FRAMEWORK 10
5.7 STANDARD OBJECTS 11
5.8 SECURITY CONSIDERATIONS 11
6. OMA DM PROTOCOL PACKAGES 12
6.1 SESSION ABORT 13
6.1.1 Description 13
6.1.2 Requirement 13
6.2 PACKAGE#0: MANAGEMENT INITIATION ALERT FROM DM SERVER TO DM CLIENT 13
6.3 PACKAGE#1: INITIALIZATION FROM DM CLIENT TO DM SERVER 13
6.4 PACKAGE#2: INITIALIZATION FROM DM SERVER TO DM CLIENT 15
6.5 PACKAGE#3: DM CLIENT RESPONSE SENT TO DM SERVER 16
6.6 PACKAGE#4: FURTHER DM SERVER MANAGEMENT OPERATIONS 17
6.7 GENERIC ALERT 17
6.7.1 Message Structure 18
6.8 DM TREE CHANGE ALERT 20
6.8.1 Message Structure 20
6.9 SESSIONLESS OPERATION 20
7. MULTIPLE MESSAGES IN PACKAGE 21
7.1 DESCRIPTION 21
7.2 REQUIREMENTS 22
8. LARGE OBJECT HANDLING 24
9. AUTHENTICATION 27
9.1 AUTHENTICATION CHALLENGE 27
9.2 AUTHORIZATION 28
9.3 APPLICATION LAYER AUTHENTICATION 28
9.4 AUTHENTICATION EXAMPLES 28
9.4.1 Basic authentication with a challenge 28
9.4.2 MD5 digest access authentication with a challenge 30
10. USER INTERACTION COMMANDS 32
10.1 INTRODUCTION 32
10.2 USER INTERACTION ALERT CODES 32
10.2.1 Display 32
10.2.2 Confirmation 33
10.2.3 User input 33
10.2.4 User choice 34

10.2.5 Progress notification (object download) 36
10.3 USER INTERACTION OPTIONS 36
10.3.1 MINDT (Minimum Display Time) 37
10.3.2 MAXDT (Maximum Display Time) 37
10.3.3 DR (Default Response) 37
10.3.4 MAXLEN (Maximum length of user input) 38
10.3.5 IT (Input Type) 38
10.3.6 ET (Echo Type) 39
11. PROTOCOL EXAMPLES 40
11.1 ONE-STEP PROTOCOL INITIATED BY THE SERVER 40
11.1.1 Package#1: Initialization from DM Client to DM Server 40
11.1.2 Package#2: Initialization from DM Server to DM Client 41
11.1.3 Package#3: DM Client response 42
11.1.4 Package#4: Acknowledgement of DM Client status 43
11.2 TWO-STEP PROTOCOL INITIATED BY THE DM SERVER 44
11.2.1 Package#1: Initialization from DM Client to DM Server 44
11.2.2 Package#2: Initialization from DM Server to DM Client 45
11.2.3 Package#3: DM Client response 47
11.2.4 Package#4: Continue with management operations 47
11.2.5 Restart protocol iteration with Package#3: DM Client response 48
11.2.6 Package#4: Finish the protocol, no continuation 49
12. BACKWARD COMPATIBILITY 50
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 51
A.1 APPROVED VERSION HISTORY 51
A.2 DRAFT/CANDIDATE VERSION 1.3 HISTORY 51
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 54
B.1 SCR FOR DM CLIENT 54
B.1.1 SCR for DM Session Setup Phase 54
B.1.2 SCR for Session Abort 55
B.1.3 SCR for Multiple Messages 55
B.1.4 SCR for Large Object 55
B.1.5 SCR for User Interaction Commands 56
B.1.6 SCR for Generic Alert 56
B.2 SCR FOR DM SERVER 57
B.2.1 SCR for DM Session Setup Phase 57
B.2.2 SCR for Session Abort 57
B.2.3 SCR for Multiple Messages 58
B.2.4 SCR for Large Object 58
B.2.5 SCR for User Interaction Commands 58
B.2.6 SCR for Generic Alert 59
APPENDIX C. PROTOCOL VALUES (NORMATIVE) 60

Figures

Figure 1: Example of how multiple messages can be used 22

1. Scope

This document describes a management protocol using the DM Representation Protocol [DMREPPRO]. This protocol is called the OMA Device Management Protocol, abbreviated as OMA DM Protocol, and it defines the protocol for various management procedures.

2. References

2.1 Normative References

[DMDICT] "OMA Device Management Dictionary, Version 1.0". Open Mobile Alliance™. OMA-SUP-DM_Dictionary-v1_0. URL:http://www.openmobilealliance.org

[DMNOTI] "OMA Device Management Notification Initiated Session, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_Notification-V1_3. URL:http://www.openmobilealliance.org

[DMREPPRO] "OMA Device Management Representation Protocol, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_RepPro-V1_3. URL:http://www.openmobilealliance.org

[DMSEC] "OMA Device Management Security, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_Security-V1_3. URL:http://www.openmobilealliance.org

[DMSTDOBJ] "OMA Device Management Standardized Objects, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_StdObj-V1_3. URL:http://www.openmobilealliance.org

[DMTND] "OMA Device Management Tree and Description, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_TND-V1_3. URL:http://www.openmobilealliance.org

[DMMETA] "Device Management Meta Information, version 1.3". Open Mobile Alliance™. OMA-TS-DM_ _MetaInfo-V1_3. 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

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

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[SLESSCMD] "OMA Device Management Sessionless Command, Version 1.3", Open Mobile Alliance™, OMA-TS-DM_Sessionless-V1_3. URL:http://www.openmobilealliance.org

[SLESSRPT] "OMA Device Management Sessionless Report, Version 1.3", Open Mobile Alliance™, OMA-TS-Sessionless Reporting-V1_3. URL:http://www.openmobilealliance.org

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

2.2 Informative References

[DMBOOT] "OMA Device Management Bootstrap, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_Bootstrap-V1_3. URL:http://www.openmobilealliance.org

[DMDDFDTD] "OMA DM Device Description Framework DTD, Version 1.3". Open Mobile Alliance™. URL:http://www.openmobilealliance.org/tech/DTD/dm_ddf-v1_3.dtd

[WBXML] "WAP Binary XML Content Format Specification", WAP Forum™, WAP-192-WBXML, 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 DTD's or XML snippets is specified in this “`typeface`”.

3.2 Definitions

Kindly consult [DMDICT] for all definitions used in this document.

3.3 Abbreviations

Kindly consult [DMDICT] for all abbreviations used in this document.

4. Introduction

The OMA DM Protocol allows management commands to be executed on nodes. It uses a package format defined in the DM Representation Protocol [DMREPPRO]. A node might reflect a set of configuration parameters for a device. Actions that can be taken against this node might include reading and setting parameter keys and values. Another node might be the run-time environment for software applications on a device. Actions that can be taken against this type of node might include installing, upgrading, or uninstalling software elements.

Actions are represented by OMA DM Protocol Commands, which are described in OMA DM Representation Protocol [DMREPPRO].

5. Protocol Overview (Informative)

OMA-DM is a secure management protocol that runs between a DM Server and a DM Client. The following sub-sections present a high-level overview of the OMA-DM protocol,

5.1 Transaction Model

The OMA-DM protocol runs within the context of a DM session, using a request/response transaction model. Once a DM session is established, the DM Server alternately sends commands to the Client and receives responses from the Client. The Client also informs the Sever about events that have occurred on the device, via Generic Alerts.

A DM session consists of two phases: the *setup phase* followed by the *management phase*. The setup phase entails authentication and device information exchange. During the management phase, the DM Server issues commands which are processed by the DM Client. The DM Client provides the status of the commands issued as well as any response that may be needed.

OMA-DM supports the notion of *Packages*. A Package is a collection of related messages that are transferred between an originator and a recipient. Generally a Package consists of a single message. However, in cases where the information to be transferred between the originator and the recipient exceeds the size limitation of a DM message, the information can be sent over multiple messages within the same Package. Each message in a Package has to be responded to individually.

DM sessions are always initiated by the DM Client. However, a Server can trigger the Client to initiate a session by sending an unsolicited message, known as the *DM Notification*, to the Client. The *DM Notification* "wakes up" the device and causes it to initiate a session with the requesting DM Server. This message can be delivered over a variety of transports including SMS, HTTP and SIP.

Normally a DM session ends when a DM Server sends an empty message (i.e. a message that does not contain any management operations or authentication challenges) to the DM Client. However, either the Client or the Server can abort the session at any time.

For more details see section 6.

5.2 DM Bootstrapping

Bootstrapping is the process by which a device moves from an un-provisioned, empty state, to a state where it is able to initiate a management session to a DM Server. A DM client that has already been bootstrapped can be further bootstrapped to additional DM Servers. Bootstrapping is a prerequisite for a DM Client to process messages from a DM Server. Bootstrapping for OMA-DM can be performed in the following ways:

- **Customized Bootstrap** (*aka* Factory Provisioning): The manufacturer includes the device management Server configuration at the time the device is manufactured. While this approach is extremely secure, it works only when the device is manufactured under contract for a specific service provider or for an enterprise customer.
- **Bootstrap From Smartcard**: All the bootstrapping information is in a Smartcard. The device obtains the bootstrap information when the Smartcard is inserted into the device.
- **Server Initiated Bootstrap**: A Bootstrap Server pushes the bootstrap information to the Client upon discovering sufficient information about the device.
- **Client Initiated Bootstrap**: The Device retrieves the bootstrap package from a Bootstrap Server, over HTTPS.

For more details see [DMBOOT].

5.3 Message Encoding Format

All OMA-DM messages, with the exception of the DM Notification message, are XML based and they conform to structure specified in [DMREPPRO]. The messages can be encoded in a tokenized, binary format defined by [WBXML]. The DM Notification message, on the other hand, is a binary message and its structure is defined in [DMNOTI].

5.4 Device Management Tree

The OMA-DM protocol supports the notion of Management Objects (MOs). These are abstract representations of remotely manageable capabilities exposed by the device. All the available MOs pertaining to a device are organized in a hierarchical tree structure known as the Management Tree. The Management Tree may be looked upon as the complete management view of a device's configuration and operational status. Different DM Servers may "see" different trees, depending upon their access rights to different portions of the Management Tree.

MOs are comprised of nodes. Each node is the potential target for invoking a management operation from the DM Server. In order to perform some remote management action, the DM Server executes an operation on the corresponding node in the DM Tree. Nodes are addressed using URIs. The URI of a node is the concatenation of the names of all the nodes from the root of the Management Tree, using '/' as the delimiter.

Each MO is characterized by a unique MOID (MO Identifier), which is generally a URN. OMA has established a well-defined procedure for the registration of MOIDs with OMNA.

For more details see [DMTND].

5.5 Protocol Commands

As mentioned in section 5.4, to carry out management actions on a device, the DM Server invokes DM commands on the various nodes in the Management Tree. The DM protocol supports the following commands that can be remotely invoked by a DM Server on a Client:

- **Get**: Retrieves the value associated with the target node in the Management Tree
- **Replace**: Sets the value associated with the target node, overwriting the previous value of the node
- **Add**: Creates a new node at the specified location in the Management Tree. The new node can be a leaf node or the root node for a management object
- **Delete**: Deletes a node, and the entire sub-tree beneath that node, if one exists
- **Exec**: Executes a predefined function, that is statically bound to the target node, on the device. Examples include initiating software download, running a diagnostic test etc.
- **Copy**: Replicates the structure and the node values associated with a sub-tree at one location to a different location within the Management Tree.

Depending upon the nature of the management command, the Client can return the status of a management command synchronously (i.e. in the response package) or asynchronously, via the Generic Alert mechanism.

A given node is likely to support only a subset of these commands.

For more details see [DMREPPRO].

5.6 Device Description Framework

Device Description Framework (DDF) is an XML-based normative format, used by devices supporting OMA-DM, to expose their manageable capabilities. These capabilities are described in terms of Management Object definitions. OMA-DM compliant MOs are required to conform to the DDF DTD, which is defined in [DMDDFDTD].

Among other things, DDF can provide the following information pertaining to each of the subtending nodes within the MO definition:

- format of data associated with the node

- MIME type of data associated with the node
- default value of data associated with the node
- allowed number of instances for the node within the sub-tree that is rooted at its parent node (i.e. indicator of how many siblings the node can have in the Management Tree at run time)
- indication whether the node is created statically or dynamically

Optionally, DDF can also be used to restrict the location of an MO in the Management Tree.

For more details see [DMTND].

5.7 Standard Objects

OMA-DM maintains complete separation of schema from protocol. Most of the MOs standardized by OMA lie outside the core DM specification. In fact, the DM Protocol has awareness of only a handful of management objects, which are referred to as "Standard Objects".

Standard Objects control the core protocol itself, rather than some domain-specific DM functionality like firmware update or device diagnostics. The Standard Objects for OMA-DM are listed below:

- **DM Account**: This MO is used to manage bootstrap settings for a DM Server. Among other things, this MO contains the Client and Server security credentials for the DM Server in question
- **Device Information**: This MO contains the basic information pertaining to the device. This information is sent to the DM Server at the beginning of each DM session.
- **Device Details**: This MO contains device specific parameters. Unlike the Device Information parameters, these parameters are sent to the DM Server only on demand.

Support for standard objects is mandatory for all OMA-DM Clients.

For more details see [DMSTDOBJ].

5.8 Security Considerations

OMA-DM is a secure protocol. Communication between the DM Server and the DM Client takes place only after a trusted relationship has been established, via the DM Bootstrap Process [DMBOOT]. OMA-DM supports multiple authentication schemes. In the case where a DM Client supports multiple authentication schemes, the DM Server can indicate the preferred authentication scheme. Both the DM Server and the DM Client can challenge each other if no credentials were given in the original request or the credentials are considered too weak.

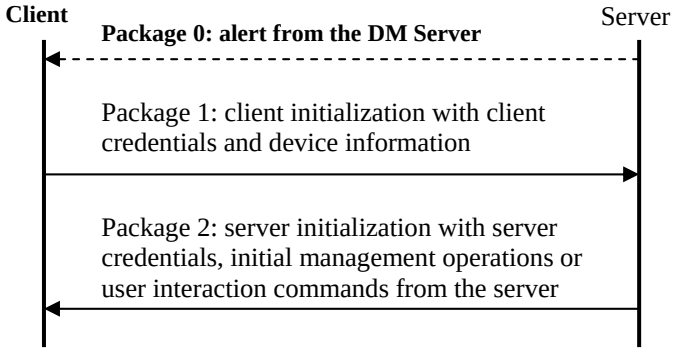
OMA-DM also supports an access control mechanism to ensure that only authorized DM Servers can invoke commands on the Management Tree.

For more details see [DMSEC] and [DMTND].

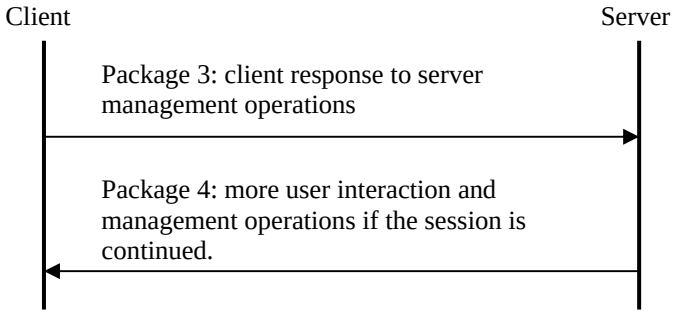
6. OMA DM Protocol packages

OMA DM Protocol consists of two parts: setup phase (authentication and device information exchange) and management phase. Management phase can be repeated as many times as the DM Server wishes. Management sessions may start with Package#0 (the trigger). Trigger may be out-of-band depending on the environment and it is specified in DM Notification Initiated Session [DMNOTI].

The following chart depicts the two phases.



Setup phase



Management phase

The Management Phase consists of a number of protocol iterations. The content of the package sent from the DM Server to the DM Client determines whether the session must be continued or not. If the DM Server sends management operations in a package that need responses (`Status` or `Results`) from the DM Client, the management phase of the protocol continues with a new package from client to DM Server containing the DM Client's responses to those management operations. The response package from DM Client starts a new protocol iteration. The DM Server can send a new management operation package and therefore initiate a new protocol iteration as many times as it wishes.

During the management phase when a package from DM Server to DM Client does not contain management operations or a challenge, the client will create a package containing only `Status` for `SyncHdr` as a response to the package received from server. In this case the entire response package MUST NOT be sent and the protocol ends. A server MUST send response packages to all client packages.

Processing of packages can consume unpredictable amount of time. Therefore the OMA DM Protocol does not specify any timeouts between packages.

If not enclosed by a Sequence or Atomic command, the client and server MAY freely choose the execution order of the management commands sent in the package. However, when execution order is required by the parent management command, commands MUST be executed in the order they were sent.

The DM Client MUST NOT send any commands other than `Replace` command containing DevInfo, `Results` and `Alert` to the DM Server.

6.1 Session Abort

6.1.1 Description

Either the client or the server may decide to abort the session at any time. Reasons for session abort may be server shutdown, client power-down, user interaction on the client, etc. In this case it is best if the aborting party sends a SESSION ABORT `Alert`. It is RECOMMENDED that the message also includes `Status` and `Results` of all the management commands that the aborting party executed before the abort operation.

If a recipient of a Session Abort sends a response to this message, the response is ignored.

Some cases of session aborts are not controllable, for example if the client goes out of coverage or its battery runs down. Servers and clients must be prepared for non-signalled session aborts as well. The requirements stated above are intended to reduce situations in which one party times out on a response from the other.

Implementations are possible (e.g. OBEX) in which the request/response roles of the transport binding may be reversed, i.e. the DM Client is a transport-level server, and the DM Server is a transport-level client. In this case, the recommendation in Section 6.1.1 above may not apply.

6.1.2 Requirement

`Alert` 1223 is used to signal an unexpected end to the device management session. The sender of the Session Abort alert MAY also include `Status` and `Results` of all the management commands that the aborting party executed before the abort operation. The sender MUST include a `Final` flag. A server receiving this alert SHOULD respond with a message that MUST contain status for the `Alert` and the `SyncHdr` and no new commands.

A client receiving `Alert` 1223 SHOULD NOT respond.

6.2 Package#0: Management Initiation Alert from DM Server to DM Client

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".

A DM Server can use this notification capability to cause the DM Client to initiate a connection back to the DM Server. OMA DM Protocol specifies several Management Initiation notification bearers. Definition of bearers and notification content can be found from [DMNOTI] specification.

Note that an identical effect to receiving a Management Initiation notification can be caused in other ways. For example, the user interface (UI) of the device may allow the user to tell the DM Client to initiate a management session. Alternatively, the DM Client might initiate a session as the result of a timer expiring. A fault of some type in the device could also cause the DM Client to initiate a session.

6.3 Package#1: Initialization from DM Client to DM Server

The purpose of the initialization package sent by the client is:

- To send the DevInfo information (like manufacturer, model, etc.) to a DM Server as specified [DMSTDOBJ]. DM Client MUST send DevInfo information in the first message of management session.
- To send any MO requested by DM Server in Package#0 message (as specified in [DMNOTI]).

- To identify the DM Client to the DM Server according to the rules specified in Section 9.
- To inform the DM Server whether the management session was initiated by the DM Server (by sending a trigger in Package#0) or by the DM Client (like end user selecting a menu item).
- To inform the DM Server of any optional Client generated alert, for example Generic Alert or Client Event [DMREPPRO].
- To optionally inform the DM Server whether the DM Tree has changed since the last session with that DM Server.

The detailed requirements for the initialization package from the DM Client to DM Server (Package#1) are:

1. The requirements for the elements within the `SyncHdr` element.
    - The value of the `VerDTD` element MUST be '1.2'.
    - The value of the `VerProto` element MUST be 'DM/1.3'.
    - `SessionID` MUST be included to indicate the ID of the management session. If the DM Client is responding to notification, with alert code SERVER-INITIATED MGMT (1200), then `SessionID` MUST be same as in notification. Note that SessionID in the Notification Message is 16 bit network byte ordered integer, and DM Client MUST convert it to the format for SessionID in DM Protocol which is a string of hexadecimal digits (chosen from the numeric digits "0"-"9" and the upper-case letters "A"-"F") of between one and four characters in length,.If the DM Client is not responding to notification, the client generates a `SessionID` which should be unique for that client. The same `SessionID` MUST be used throughout the whole session.
    - `MsgID` MUST be used to unambiguously identify the message belonging to the management session from DM Server to DM Client.
    - The `Target` element MUST be used to identify the target DM Server.
    - The `Source` element MUST be used to identify the source DM Client.
    - The `Cred` element MAY be included in the authentication message from the DM Client to DM Server as specified in Section 9.
2. `Alert` MUST be sent whether the client or the server initiated the management session in the `SyncBody`. The requirement for the `Alert` command follows:
    - `CmdID` is REQUIRED.
    - The `Data` element is used to carry the management session type which can be either SERVER-INITIATED MGMT (1200) or CLIENT-INITIATED MGMT (1201).
3. The DevInfo information MUST be sent using the `Replace` command in the `SyncBody`. The requirement for the `Replace` command follows:
    - `CmdID` is REQUIRED.
    - An `Item` element per node found from DevInfo tree. Possible nodes in DevInfo tree are specified in [DMSTDOBJ].
    - The `Source` element in the `Item` element MUST have a value indicating URI of node.
    - The `Data` element is used to carry the DevInfo data.
4. DM Client MAY include client-generated alerts such as Client Event [DMREPPRO] or Generic Alert.

5. MOs requested in the Notification message (Package#0) MUST only be sent using the Alert command in the SyncBody. All instances of the requested MO that the requesting DM Server has ACL rights to MUST be sent. The requirement for the Alert command follows:

   - `CmdID` is REQUIRED.
   - The `Data` element MUST be REQUESTED_MO (1229).
   - The `Source` element in the `Item` element MUST have a value indicating URI of the root node of the MO.
   - The `Data` element in the `Item` element MUST contain the MO in TNDS format.

   The DM Client MUST ignore the request in the Notification if no instance is found or the DM Server doesn't have proper ACL rights on any found instances.

The `Final` element MUST be used in the `SyncBody` for the message, which is the last in this package.

6.4 Package#2: Initialization from DM Server to DM Client

The purpose of the initialization package sent by the DM Server is to:

- Identify the DM Server to the DM Client according to the rules specified in Section 9.
- Optionally, the DM Server can send user interaction commands.
- Optionally to send management data and commands.
- Send status of Client Initiated Alerts if any of these was received from the DM Client

Package#2 MAY close the management session by containing only the <Final> element (any management command, user interaction command or client authentication challenge will continue the session). Alternately, the DM Server MAY send the Session Abort Alert (1223) to force the close of the session in extreme situations.

The detailed requirements for Package#2 are:

- The requirements for the elements within the `SyncHdr` element.
  - The value of the `VerDTD` element MUST be '1.2'.
  - The value of the `VerProto` element MUST be 'DM/1.3' when complying with this specification.
  - `SessionID` MUST be included to indicate the ID of the management session.
  - `MsgID` MUST be used to unambiguously identify the message belonging to the management session from DM Server to DM Client.
  - The `Target` element MUST be used to identify the target DM Client.
  - The `Source` element MUST be used to identify the source DM Server.
  - `Cred` element MAY be included in the authentication message according to the rules described in Section 9. DM Server is always authenticated to the DM Client but this authentication MAY be accomplished at the transport level.
- The `Status` MUST be returned in the `SyncBody` for the `SyncHdr` and `Alerts` sent by the DM Client.
- Any management operation including user interaction in the DM document (e.g. `Alert`, `Sequence`, `Replace`) are placed into the `SyncBody`.

- `CmdID` is REQUIRED.
- `Source` MUST be used if URI is needed to further address the source dataset.
- `Target` MUST be used if URI is needed to further address the target dataset.
- The `Data` element inside `Item` is used to include the data itself unless the command does not require a `Data` element.
- The `Meta` element inside an operation or inside an `Item` MUST be used when the `Type` or `Format` are not the default values [DMMETA].

o The `Final` element MUST be used in the `SyncBody` for the message, which is the last in this package.

6.5 Package#3: DM Client response sent to DM Server

The content of Package#3 is:

- Results of management actions sent from DM Server to DM Client.
- Results of user interaction commands.
- New optional Client generated alert, for example Generic Alert or Client Event [DMREPPRO] that was raised during the session.

This package is sent by the DM Client if Package#2 contained management commands that required a response from the DM Client.

The detailed requirements for Package#3 are:

1. The requirements for the elements within the `SyncHdr` element.
   - The value of the `VerDTD` element MUST be '1.2'.
   - The value of the `VerProto` element MUST be 'DM/1.3'.
   - `SessionID` MUST be included to indicate the ID of the management session.
   - `MsgID` MUST be used to unambiguously identify the message belonging to the management session from DM Server to DM Client.
   - The `Target` element MUST be used to identify the target DM Server.
   - The `Source` element MUST be used to identify the source DM Client.
2. `Status` MUST be returned for the `SyncHdr` and `Alert` command sent by the DM Server in the `SyncBody`.
3. `Status` MUST be returned in the `SyncBody` for management operations sent by the DM Server in Package#2.
4. `Results` MUST be returned in the `SyncBody` for successful `Get` operations sent by the DM Server in the previous package and the following requirements apply:
   - `Results` MUST contain `Meta` element with `Type` and `Format` elements describing content of `Data` element, unless the `Type` and `Format` have the default values [DMMETA].
   - Items in Results MUST contain the `Source` element that specifies the source URI.
5. Client MAY send client generated alerts, for example Client Event [DMREPPRO] or Generic Alert.

The Final element MUST be used in the SyncBody for the message, which is the last in this package.

6.6 Package#4: Further DM Server management operations

Package#4 is used to close the management session. If the DM Server sends any operation in Package#4 that needs response from the DM Client, the protocol restarts from Package#3 with a new protocol iteration. DM Server sends results of Client Initiated Alerts if any of these was received from the DM Client in previous package. The detailed requirements for Package#4 are:

1. The requirements for the elements within the SyncHdr element.
    - The value of the VerDTD element MUST be '1.2'.
    - The value of the VerProto element MUST be 'DM/1.3'.
    - SessionID MUST be included to indicate the ID of the management session.
    - MsgID MUST be used to unambiguously identify the message belonging to the management session from DM Server to DM Client.
    - The Target element MUST be used to identify the target DM Client.
    - The Source element MUST be used to identify the source DM Server.
2. Status MUST be returned for the SyncHdr sent by the DM Server in the SyncBody and if any Alerts were sent by the DM Client then the DM Server MUST send Status for these Alerts.
3. Any management operation including user interaction in the DM document (e.g. Alert, Sequence, Replace) placed into the SyncBody.
    - CmdID is REQUIRED.
    - Source MUST be used if URI is needed to further address the source dataset.
    - Target MUST be used if URI is needed to further address the target dataset.
    - The Data element inside Item is used to include the data itself unless the command does not require a Data element.
    - The Meta element inside an operation or inside an Item MUST be used when the Type or Format are not the default values [DMMETA].

The Final element MUST be used in the SyncBody for the message, which is the last in this package. Package#4 MAY close the management session by containing only the <Final> element (any management command or user interaction command will continue the session). Alternately, theDM Server may send the Session Abort Alert (1223) to force the close of the session in extreme situations.

6.7 Generic Alert

The protocol defines a Generic Alert message for Alerts generated by the DM Client that MAY have a relation to a Management Object. In the case of a relation to a Management Object then the Source and LocURI MUST identify the address to that Management Object.

Anytime after the Client or Server Initiated Management Alert, the DM Client MAY send a Generic Alert message to the DM Server. The Generic Alert message SHALL only be sent from the DM Client to the DM Server. After the DM Server has received the Generic Alert, the DM Server MUST respond with the status for how the DM Server handles all Items.

The DM Client MAY send multiple Alert messages of code “Generic Alert” or combine them together with multiple Items inside one or multiple Alert message of code “Generic Alert”. The Data in the Generic Alert message is not specified in the protocol, the protocol will specify how the DM Client can inform the DM Server what Type and Format it is. The DM Server MUST support the Generic Alert format but not all Types of the alert data. The DM Server MUST respond with status 415 “Unsupported media Type or Format” if the Type and Format are unsupported by the DM Server. If the DM Client does not support Large Object then the Alert message MUST NOT exceed the message size.

This specification only specifies what is required from the protocol perspective, some registered Generic Alerts MAY have additional requirements for different Types of Generic Alert Data. For example, one registered Type of Alert may define that a reference to a Management Object is mandatory and the Format must be of Type Integer and the Alert Data must be included inside the Data.

The optional parameter Mark MUST contain the importance level. If the parameter is omitted then the default importance level is assumed.

The DM Server MUST respond with status 200 “OK” or 202 “Accepted for processing” if the DM Server has received the Alert without any errors and is capable of processing the Data in the Alert. In other cases the DM Server MUST use one of the following error status codes: 401, 406, 407, 412, 415 or 500.

6.7.1 Message Structure

This is the basic design of a Generic Alert message:

```
<Alert>
   <CmdID>2</CmdID>
   <Data>1226</Data>      <!-- Generic Alert -->
   <Correlator>abc123</Correlator>
   <Item>
      <Source><LocURI>./SyncML/Sample</LocURI></Source>
      <Meta>
         <Type xmlns='syncml:metinf'>
            Reversed-Domain-Name: org.domain.samplealert
         </Type>
         <Format xmlns='syncml:metinf'>xml</Format>
         <Mark xmlns='syncml:metinf'>critical</Mark>   <!-- Optional -->
</Meta>
      <Data>
         <!-- Client Alert Data Goes Here -->
      </Data>
   </Item>
</Alert>
```

6.7.1.1 CmdID

This MUST be specified in the same way as all commands.

6.7.1.2 Data

This MUST be specified with the value 1226 [DMREPPRO] for Generic Alert.

6.7.1.3 Item

This is a REQUIRED parameter. Item MUST be repeated for each alert of type Generic Alert if the device will send them together inside the same Alert message.

6.7.1.4 LocURI within Source

This is an optional parameter. If the Alert is generated from a Management Object and the definition of that Management Object mandates this parameter, then it MUST be included.

6.7.1.5 Meta

The Meta element MUST be specified for the Type and Format of the Alert Data.

6.7.1.6 Type

The Type element MUST be specified and specifies the media type of the content information in the Data element. The content information for this element type MUST either be a URN as defined below or a reverse domain name. If it is a reverse domain name then the namespace identifier “Reversed-Domain-Name” MUST be specified.

The ABNF syntax for the content of URN MUST be:

alert-type = "urn:oma:at:" enabler-id ":" version ":" alert-info

enabler-id = 1*ALPHA

version = 1DIGIT "." 1DIGIT

alert-info = 1*ALPHA

The ABNF syntax for the content of reverse domain name MUST be:

reverse-domain = "Reversed-Domain-Name:" domain-part 1*("." domain-part)

domain-part = 1*(ALPHA / DIGIT)

Examples of the valid alert types are shown below:

- Content-Type: application/samplealert
- Reversed-Domain-Name: org.openmobilealliance.dm.samplealert
- urn:oma:at:dcmo:1.0:operationcomplete

6.7.1.7 Format

The Format element MUST be specified. Format MUST contain a DM identifier of the Format of the following Data element.

6.7.1.8 Mark

The Mark element MAY be specified. Mark will define the importance level of the alert message. The following levels are allowed in Generic Alert: “fatal”, “critical”, “minor”, “warning”, “informational”, “harmless” and “indeterminate”. Their order indicates the importance level with “fatal” being the most important and “indeterminate” being the least important. If the Mark element is omitted then the default importance level “informational” is assumed.

6.7.1.9 Data (inside <Item>)

The Data element MUST be specified. Data MUST use the Format and Type specified in the Meta tag.

6.7.1.10 Correlator

The Correlator is an optional field and is used when the alert is an asynchronous response to an Exec command. Typically, the Correlator field in the alert echoes the Correlator value from an Exec command and is omitted in all other instances. Registered Generic Alerts SHOULD specify how the Correlator field is used.

6.8 DM Tree Change Alert

The protocol allows two DM Tree related alerts to be sent in Package 1 - DM_TREE_UNCHANGED_ALERT (1227) and DM_TREE_CHANGED_ALERT (1228). These optional alerts are used to indicate to the DM Server whether or not changes have occurred in the DM Tree after the last session with that DM Server (e.g. user changed settings or another DM Server has connected to the DM Client and changed the DM Tree). The mechanism that tracks changes to the DM Tree SHOULD NOT be used as a trigger to start a DM Session, as that could overload the DM Server.

If there have been no changes to the DM Tree, then the DM Client MAY send an Alert with DM_TREE_UNCHANGED_ALERT (1227) in the `Data` element as part of Package 1. Note that if changes have occurred but the DM Server does not have ACL access to that data, then this alert SHALL NOT be sent.

If there have been changes to the DM Tree, then the DM Client MAY send an Alert with DM_TREE_CHANGED_ALERT (1228) in the `Data` element as part of Package#1. The `Data` element inside `Item` MUST contain the URI of the MO that contains changed data (subject to ACL rules). Note that multiple `Item` elements MAY be contained in the Alert, allowing the DM Client to indicate multiple changes in the DM Tree. Note that continuously changing data (e.g. battery status) SHOULD NOT be reported.

6.8.1 Message Structure

DM Tree Change Alert is a specially formatted Alert issued by the DM Client to indicate to the DM Server that data in the DM Tree has changed after the last session. This alert conforms to the DTD for the SyncML element *Alert*, as defined in [DMREPPRO], with the following additional constraints:

1. The *Alert* element MUST only contain the *CmdID*, *Data* and *Item* sub-elements
2. The *Data* element MUST be set to value 1228, which is the reserved value for the *DM Tree Change Alert*, as per [DMREPPRO]
3. Each occurrence of the *Item* element MUST contain only the *Source* sub-element
4. The *Source* element MUST contain only the *LocURI* sub-element and its value MUST indicate a location in the DM Tree whose value has changed

An example of the DM Tree Change Alert is given below:

```
<Alert>
      <CmdID>3</CmdID>
      <Data>1228</Data>
        <Item>
          <Source><LocURI>"/X/Y/Z"</LocURI></Source>
        </Item>
</Alert>
```

6.9 Sessionless Operation

As an optional capability, the OMA DM protocol supports the sessionless command invocation as well as the sessionless reporting features. These features are not part of the core OMA DM protocol and are not described in this document. For details see [SLESSCMD] and [SLESSRPT].

7. Multiple Messages In Package

7.1 Description

The OMA DM Protocol provides the functionality to transfer one SyncML package using multiple DM messages. This is necessary when one SyncML package is too large to be transferred in one SyncML message. For example, this limitation may be caused by the transport protocol or by the limitations of a small footprint device.

In OMA DM, the role of the package as a logical grouping of items is very limited. Most restrictions occur on messages, not on packages. For example, a command must fit entirely into one message. This includes the `Sequence` and `Atomic` commands, each of which must fit entirely into one message.

In order to avoid overwhelming a DM Client with limited resources, a DM Server is not permitted to send new commands to a client that has not yet returned a status to previous commands. In other words, most messages sent by the DM Server to the DM Client will correspond to a (one message) package, except in the case where a DM Server is sending a large object or asking for more messages (using `Alert` 1222). A package containing a large object datum will span as many messages as necessary to transmit the large object, as specified in Section 8.

Note that the DM Server is always in one of the following states with respect to package boundaries:

1. The DM Server has sent a complete package. In this state, the DM Server is awaiting status from the DM Client on the commands sent in the package. Because the status and results may be large, such as the result of `Get` commands, the DM Client may send multiple messages back to the DM Server before completing its response.
2. The DM Server has received a complete package (of responses) from the DM Client. In this state, the DM Server may send new commands to the DM Client.
3. The DM Server has sent one or more messages that are part of the same package, but has not yet sent the final message of the current package. This state is only valid when the DM Server is sending a large object, and the package will end when the last chunk of the large object is sent.

Because the underlying transports for DM messages have a request/response form, either the DM Client or the DM Server MAY be required to send a message that contains neither new commands nor a `Final` flag, in order to keep the request/response cycle going.

For example, when the DM Server is in State 1 (above), it may receive many messages from the DM Client containing `Status` and `Results`. The DM Server will respond to each such message sent by the DM Client, but may not include new commands in those responses. Messages sent by the DM Server in this state will contain a `Status` to the `SyncHdr` sent by the DM Client and also the `Alert` 1222 (More Messages). `Status` MUST BE sent in response to `Alert` but MUST NOT be sent in response to `Results`.

It is also possible for `Alert` 1222 to be replaced by `Alert` 1223 (Session Abort) if the DM Server wishes to abort the session.

The following chart shows an example of how multiple messages can be used.

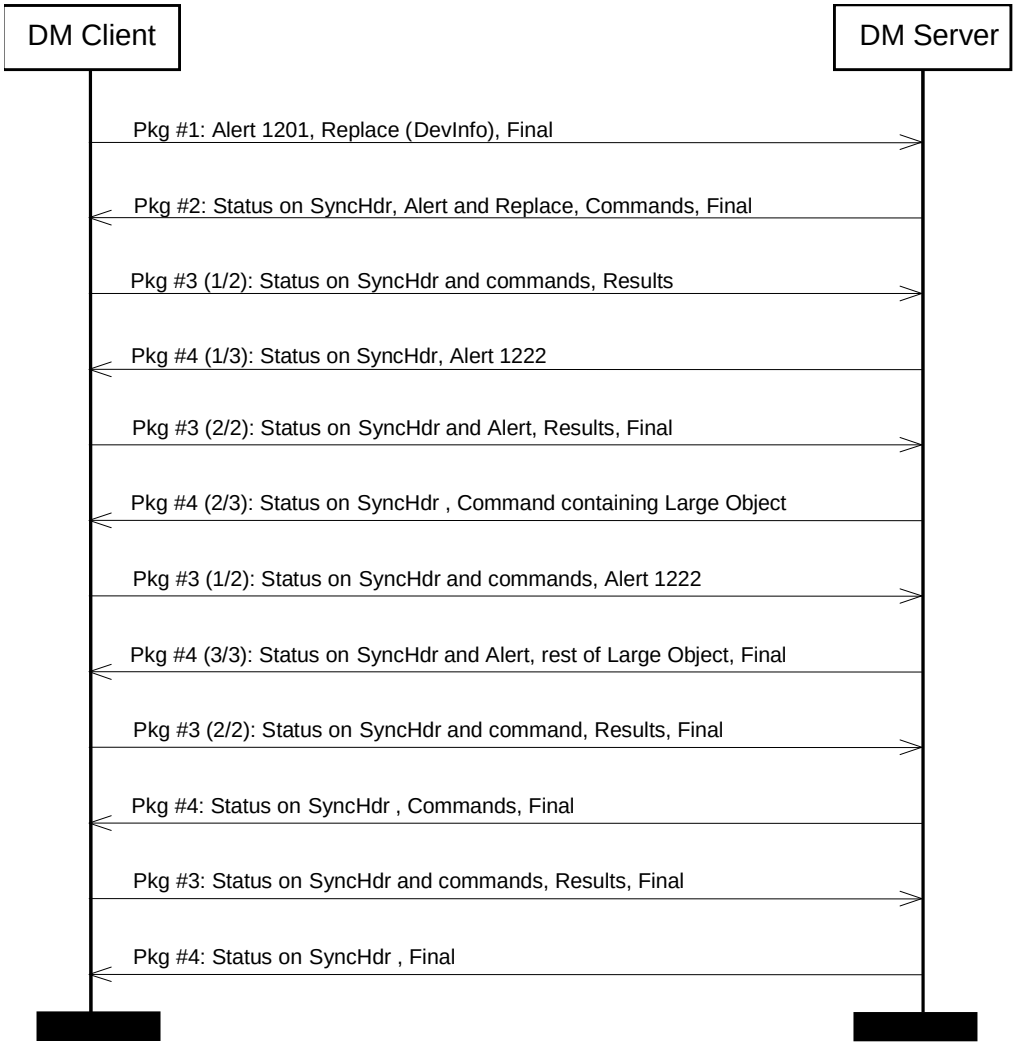


Figure 1: Example of how multiple messages can be used

7.2 Requirements

If a DM package is transferred in multiple DM messages, the last message in the package MUST include the `Final` element [DMREPPRO]. Other messages belonging to the package MUST NOT include the `Final` element.

The `Final` element MUST NOT be supplied by the DM Client to close its package until the DM Server has sent its `Final` element to close the previous package. For instance, the DM Client MUST NOT supply the `Final` element to close Package #2 or Package #4 until the DM Server has supplied the `Final` element which closes the previous package (#1 or #3, respectively). This is necessary because Packages #2 and #4 constitute replies to the commands in Packages #1 and #3.

The recipient of a DM package containing multiple messages MUST be able to ask for more messages. This is done by sending an `Alert` command, with the alert code 1222, back to the sender. If there are DM commands to be sent as a response to a preceding message, i.e. `Results`, the `Alert` command with the 1222 alert code MAY be omitted.

In the situation in which the DM Server has sent the `Final` flag, and the DM Client has not yet sent its `Final` flag, the DM Server MUST respond to the DM Client with the following "Next Message" response:

The "Next Message" response contains `Alert` code 1222 (or 1223 to abort), status to the `SyncHdr`, no other commands, and no `Final` flag.

A DM Server MUST send the `Final` flag in every message, when possible. This is not possible during the sending of a Large Object (see Section 8), or when sending the "Next Message" response.

8. Large Object Handling

The protocol provides a means to synchronize an object whose size exceeds that which can be transmitted within one message. This is achieved by splitting the object into chunks that will fit within the message and using the <MoreData/> element to signal to the recipient that the data item is incomplete and has further chunks to come.

DM Clients SHOULD support Large Objects and DM Servers MUST support Large Objects.

On receipt of a data object with the <MoreData/> element, the recipient MUST respond with a status response “213 – Chunked item accepted and buffered” and, if there are no other commands to be sent, ask for the next message using the Alert 1222 mechanism defined in Section 7.

On receipt of the last chunk of the data object the recipient reconstructs the data object from its constituent chunks and applies the requested command. The appropriate status MUST then be sent to the originator. A command on a chunked object MUST implicitly be treated as atomic; i.e. the recipient can only commit the object once all chunks have been successfully received and reassembled.

Data objects that fit within a single message MUST NOT be followed by the <MoreData/> element. Data objects that span multiple messages MUST have the <MoreData/> element after all chunks except the last chunk.

A new data object MUST NOT be added by a sender to any message until the previous data object has been completed. If a data object is chunked across multiple messages the chunks MUST be sent in contiguous messages. New DM commands (i.e. Add, Replace, Delete, Copy, Atomic or Sequence) or new Items MUST NOT be placed between chunks of a data object.

Meta and Item information SHOULD be repeated on each subsequent message containing chunks of the same data object. Authentication details related to the data object MAY vary between messages bearing chunks of the same data object as defined in the Section 9.

DM Clients that support Large Object Handling MUST indicate this by having the value of the DevDetail/LrgObj flag set to "true". The `MaxObjSize` accepted by the sender MAY be included in Meta information for the message header (`SyncHdr`) sent to the other party. `MaxObjSize` information sent in Meta information for the `SyncHdr` MUST be respected by the recipient, who MUST NOT send any single object larger than this size. If `MaxObjSize` is not sent, the recipient is free to send objects of any size back to the sender.

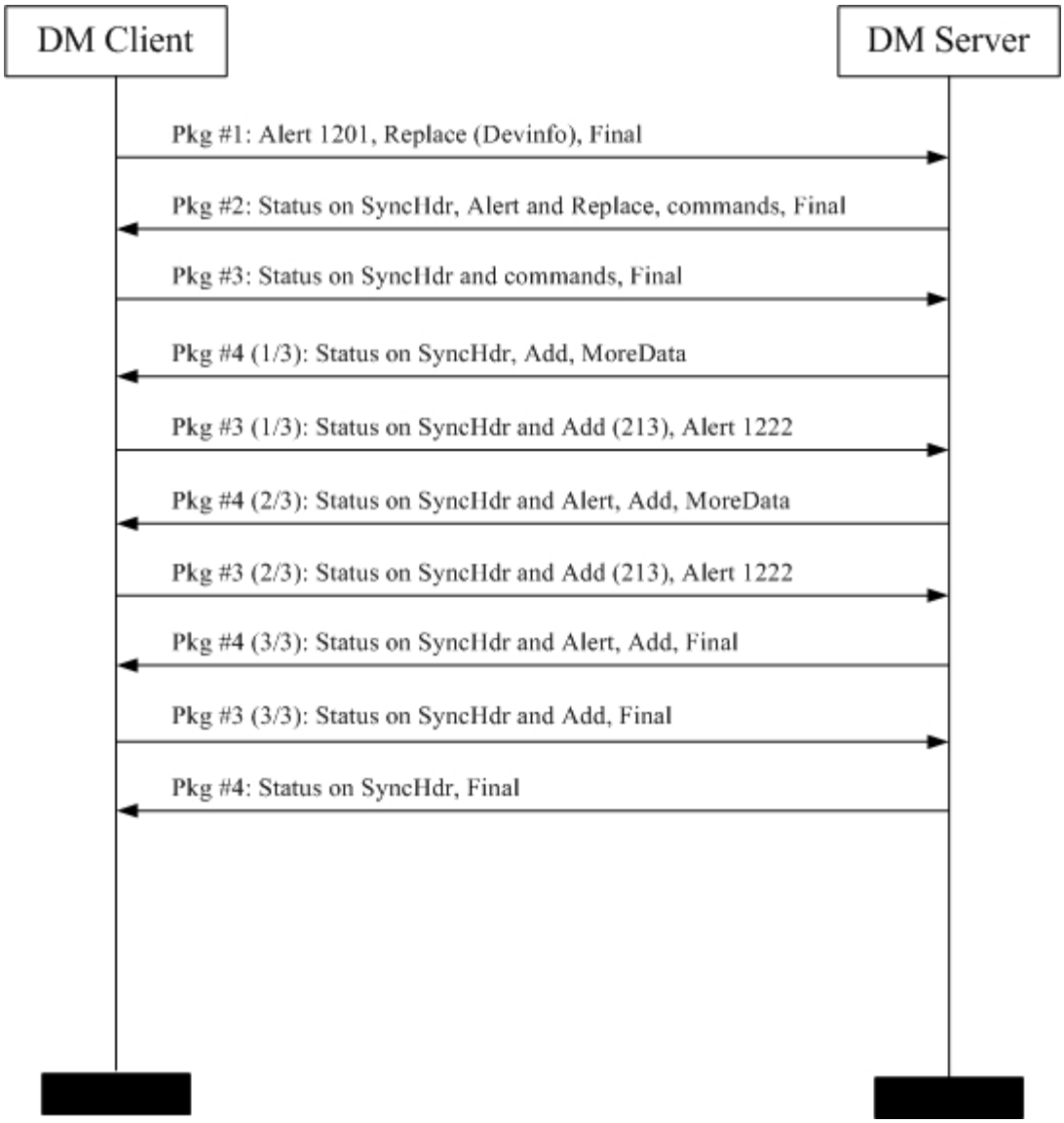
Note that the `MaxObjSize` remains in effect for the entire DM Session, unless a new value is supplied in a subsequent message. A possible reason to send a new `MaxObjSize` in a later message in the same session might be that the `MaxObjSize` of a client might depend on free memory, which can decrease as objects are created and increase as objects are deleted. The `MaxObjSize` need not be a dynamic quantity, however.

If an item is chunked across multiple messages, the <Size> element of the Meta information MUST be used to signal to the recipient the overall size of the data object. The <Size> element MUST only be specified in the first chunk of the item.

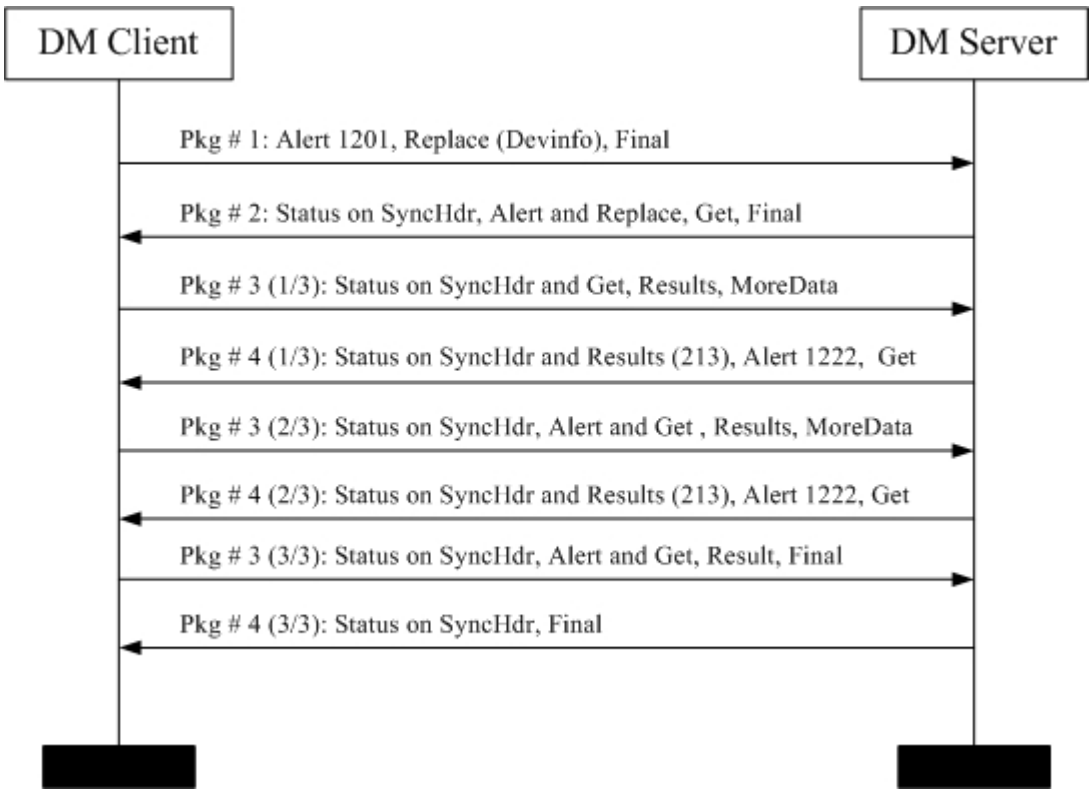
On receipt of the last chunk, the recipient MUST validate that the size of re-constituted chunks match the object <Size> supplied in the Meta information by the sender. If the size does not match then error status 424 “Size mismatch” MUST be returned. The recipient MUST NOT commit the command. The sender MAY attempt to retransmit the entire data object.

If the recipient detects a new data object or command before the previous item has been completed (by the chunk without the <MoreData/> Element), the recipient MUST respond with an Alert 1225 “End of Data for chunked object not received”. The Alert SHOULD contain the source and/or target information from the original command to enable the sender to identify the failed command. Note: a Status would not suffice here because there would not necessarily be a command ID to refer to. The recipient MUST NOT commit the command. The sender MAY attempt to retransmit the entire data object.

The following charts show examples of how Large Object handling can be used.



Example of Add a Large Object



Example of Get a Large Object

9. Authentication

OMA DM Protocol uses the authentication framework specified in this chapter, with extensions defined in OMA Device Management Security [DMSEC]. This section specifies the rules for how the OMA-DM Protocol-level and the transport-level authentication are used.

DM Server and DM Client can both challenge each other if no credentials were given in the original request or the credentials were considered too weak. If the DM Server sent no credentials or invalid credentials in Package #2, no challenge and no commands (only Status to SyncHdr and DevInfo), the DM Client MUST NOT challenge the DM Server by sending back only a Status for the SyncHdr with a challenge. If the DM Server challenged the DM Client in Package#2, the DM Client MUST revert to Package#1 and MUST resend the Alert and DevInfo along with the credentials requested by the DM Server.

The preferred authentication type of the DM Server MAY be indicated to the DM Client using the <X>/AAuthPref parameter in DM Account management object [DMSTDOBJ].

Generation and maintenance of DM Client and DM Server credentials are out of scope of the OMA DM Protocol specification.

In this chapter, the authentication procedures are defined for the basic and MD5 digest access authentication.

9.1 Authentication Challenge

If the response code to a request (message or command) is 401 (‘Unauthorized’) or 407 (‘Authentication required’), the request requires authentication. In this case, the Status command to the request MUST include a Chal element (See [DMREPPRO]). The Chal contains a challenge applicable to the requested resource. The originator MAY repeat the request with a suitable Cred element (See [DMREPPRO]). If the request already included the Cred element, then the 401 response indicates that authorization has been refused for those credentials.

Both the DM Client and the DM Server can challenge for authentication.

If the 407 response (i.e., Status) contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the entity that was given in the response, since that entity might include relevant diagnostic information.

If the response code to a request is 212 (‘Authentication accepted’), no further authentication is needed for the remainder of the DM session. In the case of the SHA256 Digest and MD5 Digest access authentication, the Chal element can however be returned. Then, the next nonce in Chal MUST used for the digest when the next DM session is started.

If a request includes security credentials and the response code to the request is 200, the same credentials MUST be sent within the next request. If the Chal element is included and the MD5 digest access authentication is mandated, a new digest is created by using the next nonce. In the case of the SHA256 Digest and MD5 Digest access authentication, the Chal element can however be returned. The next nonce in Chal MUST be used when the next request is sent.

Once authentication has occurred, the authentication type for a security layer MUST be kept same for the whole session.

In case of authentication failure (either the credentials were wrong or authentication was mandated) requirements are:

- The response message indicating the authentication failure on application layer (see chapter 9.3) contains only Status commands (i.e. Replace, Get etc. commands MUST NOT be specified in the response). A Status command MUST be provided for every command received in the request.
- In case the session is continued, the next message containing the proper credentials MUST contain a Status for the SyncHdr, MUST have the same SessionID as the previous messages and the message MUST be sent to the RespURI, if it was specified in the response indicating the authentication failure.

9.2 Authorization

The Cred element MUST be included in requests (message or command), which are sent after receiving the 401 or 407 responses if the request is repeated. In addition, it can be sent in the first request from a DM Client if the authentication is mandated through pre-configuration. The content of the Cred element is specified in [DMREPPRO]. The authentication type is dependent on the challenge (See the previous chapter) or the pre-configuration.

9.3 Application Layer Authentication

The authentication on the application layer is accomplished by using the Cred element in SyncHdr and the Status command associated with SyncHdr. Within the Status command, the challenge for the authentication is carried as defined earlier. The authentication can happen both directions, i.e., the DM Client can authenticate itself to the DM Server and the DM Server can authenticate itself to the DM Client.

9.4 Authentication Examples

9.4.1 Basic authentication with a challenge

At this example, the DM Client tries to initiate with the DM Server without any credentials (Package #1). The DM Server challenges the DM Client (Package #2) for the application layer authentication. The DM Client MUST send Package #1 again with the credentials. The DM Server accepts the credentials and the session is authenticated (Package #2). In the example, commands in SyncBody are not shown although in practice, they would be there.

Package #1 from DM Client:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target><LocURI>http://www.syncml.org/mgmt-server</LocURI></Target>
    <Source><LocURI>IMEI:493005100592800</LocURI></Source>
  </SyncHdr>
  <SyncBody>
    ...
  </SyncBody>
</SyncML>
```

Package #2 from DM Server:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target><LocURI>IMEI:493005100592800</LocURI></Target>
    <Source><LocURI>http://www.syncml.org/mgmt-server</LocURI></Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>0</CmdRef>
```

```
        <Cmd>SyncHdr</Cmd>
        <TargetRef>http://www.syncml.org/mgmt-server</TargetRef>
        <SourceRef>IMEI:493005100592800</SourceRef>
        <Chal>
          <Meta>
            <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
            <Format xmlns='syncml:metinf'>b64</Format>
          </Meta>
        </Chal>
        <Data>407</Data> <!-- Credentials missing -->
      </Status>
      ...
    </SyncBody>
</SyncML>
```

Package #1 (with credentials) from DM Client:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target><LocURI>http://www.syncml.org/mgmt-server</LocURI></Target>
    <Source><LocURI>IMEI:493005100592800</LocURI></Source>
    <Cred>
      <Meta>
      <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
      <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>QnJ1Y2UyOk9oQmVoYXZl</Data>
      <!-- base64 formatting of ‘userid:password’ -->
    </Cred>
  </SyncHdr>
  <SyncBody>
    ...
  </SyncBody>
</SyncML>
```

Package #2 from DM Server:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target><LocURI>IMEI:493005100592800</LocURI></Target>
    <Source><LocURI>http://www.syncml.org/mgmt-server</LocURI></Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>2</MsgRef><CmdRef>0</CmdRef><Cmd>SyncHdr</Cmd>
      <TargetRef>http://www.syncml.org/mgmt-server</TargetRef>
      <SourceRef>IMEI:493005100592800</SourceRef>
```

```
        <Data>212</Data> <!-- Authenticated for session -->
      </Status>
    </SyncBody>
</SyncML>
```

9.4.2 MD5 digest access authentication with a challenge

At this example, assume (as in 9.4.1 above) the DM Client tries to initiate with the DM Server without any credentials (Package #1 is omitted here for brevity). The DM Server challenges the DM Client as above (Package #2 is also omitted from this example) for the application layer authentication. The authentication type is now syncml:auth-md5 (MD5 digest access authentication). The DM Client MUST resend Package #1 this time with the MD5 credentials (as shown below in Package #1). The DM Server accepts the credentials and the session is authenticated (Package #2 below). Also, the DM Server sends the next nonce to the DM Client, which the DM Client MUST use when the next DM session is started. In the example, commands in SyncBody are not shown although in practice, they would be there.

Package #1 from DM Client:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target><LocURI>http://www.syncml.org/mgmt-server</LocURI></Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    <LocName>Bruce2</LocName>    <!-- userid -->
    </Source>
    <Cred>
      <Meta>
       <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
       <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
    <!-- Base64 coded MD5 for user ‘Bruce2’, password ‘OhBehave’, nonce
    ‘Nonce’ -->
    </Cred>
  </SyncHdr>
  <SyncBody>
    ...
  </SyncBody>
</SyncML>
```

Package #2 from DM Server:

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target><LocURI>IMEI:493005100592800</LocURI></Target>
    <Source><LocURI>http://www.syncml.org/mgmt-server</LocURI></Source>
  </SyncHdr>
  <SyncBody>
```

```
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>0</CmdRef>
      <Cmd>SyncHdr</Cmd>
      <TargetRef>http://www.syncml.org/mgmt-server</TargetRef>
      <SourceRef>IMEI:493005100592800</SourceRef>
      <Chal>
        <Meta>
          <Type xmlns='syncml:metinf'>syncml:auth-md5</Type>
          <Format xmlns='syncml:metinf'>b64</Format>
          <NextNonce xmlns='syncml:metinf'>LG3iZQhhdmKNHg==</NextNonce>
          <!-- This nonce is used at the next session -->
        </Meta>
      </Chal>
      <Data>212</Data> <!-- Authenticated for session -->
    </Status>
  </SyncBody>
</SyncML>
```

10.User interaction commands

10.1 Introduction

The OMA DM Protocol specifies following user interaction types for example to notify and obtain confirmation from the user regarding the management operation. These interaction types are the following:

- User displayable notification associated with a certain action.
- Confirmation from the user to execute a certain management operation.
- Prompt user to provide input for upcoming management operation.
- Prompt user to select item or items among items.
- Display progress notification for a certain action.

10.2 User interaction alert codes

These `Alert`s can be sent only from the DM Server to the DM Client. Clients MUST report 406: “Optional Feature Not Supported”, if DM Client does not support the specific User Interaction Alerts type. If sent by the DM Client, they are ignored by the DM Server. Multiple user interaction `Alert`'s can be present in Package#2, in this case the DM Client executes them by arbitrary order (unless `Sequence` is used) and sends back the results in multiple `Status` packages in Package#3. If the protocol continues after Package#4, Package#4 can also contain user interaction `Alert`'s.

When a user interaction is executed, DM Server is notified about the outcome of the interaction in a `Status` message. The user interaction-specific `Status` responses are described in [DMREPPRO].

All user interaction `Alert`s contain two or more `Item` elements. DM Client MUST preserve the order of these `Item` elements. DM Client MUST also process these `Item` elements in the same order as they are in the message.

User interactions, except display, SHOULD have user option to cancel operation. If the user decides to cancel the operation, then management message processing is stopped. Status codes for executed commands are reported normally and status code `(215) Not executed` is returned to all commands which are not processed. After processing the user response the DM Server might decide to continue protocol with some other management operation.

If the UI allows the user to cancel (for any of the UI `Alerts`), then the status `(214) Operation cancelled` should be returned for the `Alert`.

10.2.1 Display

The DM DISPLAY `Alert` is slightly changed in OMA DM Protocol. The `Alert` has two Items.

- The first `Item` contains optional parameters as specified in Section 10.3.
- The second `Item` has exactly one `Data` element containing the text to be displayed to the user.

Example:

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1100</Data>
  <Item><Data>MINDT=10</Data></Item>
  <Item>
    <Data>Management in progress</Data>
  </Item>
</Alert>
```

10.2.2 Confirmation

Confirmation is a binary decision: the user either approves or rejects the option. A new `Alert` code is introduced for this purpose, the CONFIRM_OR_REJECT. When the DM Client receives this `Alert`, it displays the `Alert` text then enables the user to select "Yes" or "No". If the answer is "Yes", the status code 200: “Yes” is returned and the processing of the package continues without change in processing. If the answer is "No", the status code 304: “No” is returned and the processing of the package ceases. If the UI allows the user to cancel, status 214: “Operation cancelled” should be returned for the Alert.

If user answers "No", then package processing will change according to placement of confirmation `Alert` in package as follows.

- If confirmation `Alert` is inside `Atomic`, then `Atomic` fails and all executed commands have to be rolled back.
- If confirmation `Alert` is inside `Sequence`, then commands in `Sequence` after confirmation `Alert` are bypassed.
- If confirmation `Alert` is not inside `Atomic` or `Sequence` i.e. it is directly in `SyncBody`, then user response has no effect to package processing. In this way DM Server can query user opinion before sending actual management commands to DM Client.

Status code `(215) Not Executed` will be sent back for the commands whose execution was bypassed as result of user interaction.

The `Alert` contains two `Item`s.

- The first `Item` contains the optional parameters as specified in Section 10.3.
- The second `Item` has exactly one `Data` element containing the text to be displayed to the user.

Example:

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1101</Data>
  <Item></Item> <!-- no optional parameters -->
  <Item>
    <Data>Do you want to add the CNN access point?</Data>
  </Item>
</Alert>
```

Result if user responds "No":

```
<Status>
  <CmdID>2</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>2</CmdRef>
  <Cmd>Alert</Cmd>
  <Data>304</Data> <!-- Answer was “no” -->
</Status>
```

If the result in the above example had been that the user chose Yes, the status would have been `(200)`.

10.2.3 User input

When this `Alert` is sent, the client displays the text then allows the user to type in a text string. This text string is then sent back to the server in a `Status` message.

The DM Server instructs the DM Client to execute this user interaction by sending a TEXT INPUT `Alert`. The `Alert` contains at least two `Item`s.

- The first `Item` contains optional parameters as specified in Section 10.3.
- The second `Item` has exactly one `Data` element containing the text to be displayed to the user.

Example:

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1102</Data>
  <Item></Item>
  <Item>
    <Data>Type in the name of the service you would like to
configure</Data>
  </Item>
</Alert>
```

The user is presented with the text and an input box to type in the message. The following `Status` message is sent back in the next message from DM Client to DM Server:

```
<Status>
  <MsgRef>1</MsgRef>
  <CmdRef>2</CmdRef>
  <Cmd>Alert</Cmd>
  <Data>200</Data> <!-- Successful, user typed in a text -->
  <Item>
    <Data>CNN</Data> <!-- User input -->
  </Item>
</Status>
```

10.2.4 User choice

When this `Alert` is sent, the user is presented with a set of possible choices. The `Alert` body MUST contain the following `Item`s.

- The first `Item` contains optional parameters as specified in Section 10.3.
- The second `Item` has exactly one `Data` element containing the title of the selection as plain text.
- From third `Item` onwards the Item contains exactly one `Data` element that describes one possible choice as plain text. These `Item`s are referenced by a number starting from 1. `Item`s MUST be numbered in the order they were sent. `Item`s SHOULD be presented to the user in the order they were sent.

The user selection is returned in `Status` message. The selected item is returned in an `Item`. The `Data` element of this `Item` contains the reference number of item. A variation of this `Alert` allows the user to select multiple items. In this case selected items are sent back in multiple `Item`s in the same way as one selected item.

One possible implementation could be a list and each `Data` member of the `Alert` could be displayed as a row in the list. The user could select a list item then he or she would push the "Ok" button and the ID of the selected list item is sent back in a `Status` message.

Example for a single-choice `Alert`:

```
<Alert>
```

```
   <CmdID>2</CmdID>
   <Data>1103</Data>
   <Item><Data>MINDT=10</Data></Item>
   <Item>
     <Data>Select service to configure</Data>
   </Item>
   <Item>
     <Data>CNN</Data>
   </Item>
   <Item>
     <Data>Mobilbank</Data>
   </Item>
   <Item>
     <Data>Game Channel</Data>
   </Item>
</Alert>
```

Response to this `Alert` returns the selected item.

```
<Status>
   <MsgRef>1</MsgRef>
   <CmdRef>2</CmdRef>
   <Cmd>Alert</Cmd>
   <Data>200</Data> <!-- Successful, user selected an item -->
   <Item>
     <Data>2</Data> <!-- User selected MobilBank -->
   </Item>
</Status>
```

Example to multiple-choice `Alert`

```
<Alert>
   <CmdID>2</CmdID>
   <Data>1104</Data>
   <Item></Item>
   <Item><Data>Select service to configure</Data></Item>
   <Item>
     <Data>CNN</Data>
   </Item>
   <Item>
     <Data>Mobilbank</Data>
   </Item>
   <Item>
     <Data>Game Channel</Data>
   </Item>
</Alert>
```

Response to this `Alert` returns the selected item. The number of the selected items can be returned in arbitrary order by the DM Client.

```
<Status>
   <MsgRef>1</MsgRef>
   <CmdRef>2</CmdRef>
   <Cmd>Alert</Cmd>
```

```
  <Data>200</Data> <!-- Successful, user selected an item -->
  <Item>
    <Data>3</Data>
  </Item>
  <Item>
    <Data>2</Data> <!-- User selected Mobilbank and Game Channel -->
  </Item>
</Status>
```

10.2.5 Progress notification (object download)

Users SHOULD be able to track the progress of a longer management operation like a file or object download. OMA DM Protocol will not provide a separate mechanism for progress notification but it will entirely reuse the DM `Size` Meta-Information tag defined in DM Meta-Information DTD [DMMETA] and will make a recommendation for device manufacturers to use this tag for displaying progress notification.

According to DM Meta-Information DTD, any `Item` can be tagged by `Size` meta-information that indicates the size of the object. When the DM Client encounters a `Size` meta-information tag in a received `Item`, it MAY display a progress notification on the user interface if the DM Client decides that the item with the given size will take a longer time to download. The progress notification bar is scaled according to the length information conveyed in the `Size` element. If the size information is not sent by the DM Server, the DM Client is not able to display a scaled progress bar so it is recommended that DM Servers send this information if the object to be downloaded by the DM Client is reasonably large.

Example of an antivirus data file download with `Size` Meta Information.

```
<Add>
  <CmdID>2</CmdID>
  <Meta>
    <Format xmlns='syncml:metinf'>b64</Format>
    <Type xmlns='syncml:metinf'>
      application/antivirus-inc.virusdef
    </Type>
  </Meta>
  <Item>
    <Meta>
      <!-- Size of the data item to download -->
      <Size xmlns='syncml:metinf'>37214</Size>
    </Meta>
    <Target><LocURI>./antivirus_data</LocURI></Target>
    <Data>
      <!-- Base64-coded antivirus file -->
    </Data>
  </Item>
</Add>
```

Progress indicator will be displayed during the execution of the `Add` command and it will be scaled so that the total length of data to be downloaded is supposed to be 37214 bytes.

10.3 User interaction options

`Alert`'s MAY have optional User interaction parameters in the first `Item`. Optional parameters are represented as one text string inside the `Data` element. If the User interaction `Alert` does not have optional parameters, the first `Item` is empty. The optional parameter string conforms to the URL encoding format specified in [RFC2396].

The following example uses two optional parameters:

```
MAXDT=30&DR=1
```

The DM Client MUST skip without error message all the optional parameters that it is not able to process.

The following optional parameters are currently defined.

10.3.1 MINDT (Minimum Display Time)

This parameter is a hint to the user agent of the minimum time that the user interaction should be displayed to the user. This can be important to guarantee that a notification message is readable.

MINDT parameter MUST have a value that can be evaluated as a positive, integer number. Value of MINDT is interpreted as notification display time to user in seconds.

Example:

```
<!-- Display this message for at least 10 seconds -->
<Item><Data>MINDT=10</Data></Item>
```

10.3.2 MAXDT (Maximum Display Time)

This parameter is a hint to the user agent for how long the DM Client should wait for the user to execute the user interaction. If the user does not act within MAXDT time, the action is considered to be cancelled and a timeout status package or default response package is sent back to the DM Server.

MAXDT parameter MUST have a value that can be evaluated as a positive, integer number. Value of MAXDT is interpreted as seconds to wait for user action.

Example:

```
<!-- Wait maximum 20 seconds for the user -->
<Item><Data>MAXDT=20</Data></Item>
```

10.3.3 DR (Default Response)

DR optional parameter specifies the initial state of the user interaction control widget. Other than setting the initial state of the user interaction control widget, DR has no other influence on the user interaction control widget. Interpretation for different user interaction types is the following:

- If the user interaction is Notification, this optional parameter is ignored.
- If the user interaction is a confirmation, 0 means that the reject user interface element is highlighted by default, 1 means that the accept user interface element is highlighted by default. Highlighted user interface element means that the "default" user interaction (like pressing Enter button) will select the highlighted user interface element. If the DM Client user interface has no notion of highlighted user interface element, this parameter MAY be ignored.
- If the user interaction is user input, DR value specifies the original text in the text input user interface element. This text MUST conform to the optional parameter syntax rules.
- If the user interaction is single-choice, the DR value is the originally highlighted choice item; e.g. value between 1 and the number of items in the selection list.
- If the user interaction is a multi-choice, the DR value is a minus sign-separated list of originally highlighted values (for example: 2-3).

Examples:

```
<!-- Accept by default in a Confirmation action -->
<Item><Data>DR=1</Data></Item>
```

```
<!-- Default user entry of ‘John Doe’ in an user input action -->
<Item><Data>DR=John+Doe</Data></Item>
```

```
<!-- Default selection of item 3 in a single-choice action -->
<Item><Data>DR=3</Data></Item>
```

```
<!-- Default selection of item 2 and 3 in a multi-choice action -->
<Item><Data>DR=2-3</Data></Item>
```

10.3.4 MAXLEN (Maximum length of user input)

MAXLEN value is evaluated to a positive integer and determines the maximum number of characters that can be typed into the text input user interaction widget. The optional parameter MUST be ignored in all other kind of user interaction widget. If the specified maximum length of input string exceeds the capability of the DM Client, the client MAY ignore the parameter.

Example:

```
<!-- Maximum string length is 30 -->
<Item><Data>MAXLEN=30</Data></Item>
```

10.3.5 IT (Input Type)

IT specifies what kind of characters is allowed in the text input user interaction widget. Based on this information a DM Client with limited keyboard MAY display user interaction elements that allow easy input of characters not present on the keyboard. The optional parameter MUST be ignored in user interaction widgets other than text input. Allowed values:

IT=A - Alphanumeric input, DM Client SHOULD allow input of all alphanumeric characters. This is the default behaviour.

IT=N - Numeric input, DM Client SHOULD allow input of all numeric characters, decimal point and sign character.

IT=D - Date input, DM Client SHOULD allow input of all numeric characters. User input is delivered to DM Server in following text string format "DDMMYYYY", where;

- DD is day with possible leading zero.
- MM is month with possible leading zero.
- YYYY is year presented with four digits.

IT=T - Time input, DM Client SHOULD allow input of all numeric characters. User input is delivered to DM Server in following text string format "hhmmss", where;

- hh is hours with possible leading zero.
- mm is minutes with possible leading zero.
- ss is seconds with possible leading zero.

IT=P - Phone number input, DM Client SHOULD allow input of all numeric characters, "+", "p", "w" and "s". "+" MUST be first if present in phone number.

IT=I - IP address input, DM Client SHOULD allow input of all numeric characters. User input is delivered to DM Server in following text string format "xxx.yyy.zzz.www"

Example:

```
<!-- Numeric text input -->
<Item><Data>IT=N</Data></Item>
```

`Status` message delivered to DM Server as response

```
<Status>
  <MsgRef>1</MsgRef>
  <CmdRef>2</CmdRef>
  <Cmd>Alert</Cmd>
  <Data>200</Data> <!-- Successful, entered a number -->
  <Item>
    <Data>-1.23</Data>
  </Item>
</Status>
```

10.3.6 ET (Echo Type)

ET specifies how text input user interaction widget echoes the characters that the user types in. The optional parameter MUST be ignored in user interaction widgets other than text input. Allowed values:

ET=T - Text input. The DM Client SHOULD allow the user to see the character the user typed into the text input user interaction widget. This is the default behaviour.

ET=P - Password input. The DM Client SHOULD hide the character the user typed into the text input user interaction widget. One way of doing it MAY be writing an asterisk instead of the character itself.

Example:

```
<!-- Numeric text input -->
<Item><Data>ET=T</Data></Item>
```

11.Protocol examples

In this section several protocol scenarios will be demonstrated.

11.1 One-step protocol initiated by the server

In this section an example is presented in which a WAP connectivity context is added to the WAP settings. The user is asked to confirm whether the settings could be added.

11.1.1 Package#1: Initialization from DM Client to DM Server

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
    <Cred> <!-- Client credentials are mandatory if the transport layer is
    not providing authentication.-->
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data>
        <!-- base64 formatting of userid:password -->
      </Data>
    </Cred>
    <Meta> <!-- Maximum message size for the client -->
      <MaxMsgSize xmlns=’syncml:metinf’>5000</MaxMsgSize>
    </Meta>
  </SyncHdr>
  <SyncBody>
    <Alert>
      <CmdID>1</CmdID>
      <Data>1200</Data> <!-- Server-initiated session -->
    </Alert>
    <Replace>
      <CmdID>3</CmdID>
      <Item>
        <Source><LocURI>./DevInfo/DevId</LocURI></Source>
        <Meta>
          <Format xmlns='syncml:metinf'>chr</Format>
          <Type xmlns='syncml:metinf'>text/plain</Type>
        </Meta>
        <Data>IMEI:493005100592800</Data>
      </Item>
      <Item>
        <Source><LocURI>./DevInfo/Man</LocURI></Source>
        <Meta>
          <Format xmlns='syncml:metinf'>chr</Format>
          <Type xmlns='syncml:metinf'>text/plain</Type>
```

```
          </Meta>
          <Data>Device Factory, Inc.</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/Mod</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>SmartPhone2000</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/DmV</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>1.0.0.1</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/Lang</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>en-US</Data>
        </Item>
      </Replace>
      <Final/>
    </SyncBody>
</SyncML>
```

11.1.2 Package#2: Initialization from DM Server to DM Client

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>IMEI:493005100592800</LocURI>
    </Target>
    <Source>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Source>
    <Cred> <!-- Server credentials -->
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data><!-- base64 formatting of userid:password --></Data>
    </Cred>
  </SyncHdr>
  <SyncBody>
    <Status>
```

```
        <MsgRef>1</MsgRef><CmdRef>0</CmdRef>
        <Cmd>SyncHdr</Cmd>
        <CmdID>6</CmdID>
        <TargetRef>http://www.syncml.org/mgmt-server</TargetRef>
        <SourceRef>IMEI:493005100592800</SourceRef>
        <!-- Authenticated for the session -->
        <Data>212</Data>
      </Status>
      <Status>
        <MsgRef>1</MsgRef><CmdRef>1</CmdRef>
        <CmdID>7</CmdID>
        <Cmd>Alert</Cmd>
        <Data>200</Data><!-- OK -->
      </Status>
      <Status>
        <MsgRef>1</MsgRef><CmdRef>3</CmdRef>
        <CmdID>8</CmdID>
        <Cmd>Replace</Cmd>
        <Data>200</Data><!-- OK -->
      </Status>
      <Sequence>
        <CmdID>1</CmdID>
        <Alert>
          <CmdID>2</CmdID>
          <Data>1101</Data> <!-- User confirmation required -->
          <Item></Item>
          <Item>
            <Data>Do you want to add the CNN access point?</Data>
          </Item>
        </Alert>
        <Replace>
          <CmdID>4</CmdID>
          <Meta>
            <Format xmlns='syncml:metinf'>b64</Format>
            <Type xmlns='syncml:metinf'>
              application/vnd.wap.connectivity-wbxml
            </Type>
          </Meta>
          <Item>
            <!-- CNN WAP settings object in the settings -->
            <Target>
              <LocURI>./settings/wap_settings/CNN</LocURI>
            </Target>
            <Data><!-- Base64-coded WAP connectivity document --></Data>
          </Item>
        </Replace>
      </Sequence>
      <Final/>
    </SyncBody>
</SyncML>
```

11.1.3 Package#3: DM Client response

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

```
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdID>1</CmdID>
      <CmdRef>0</CmdRef>
      <Cmd>SyncHdr</Cmd>
      <!-- SyncHdr accepted -->
      <Data>212</Data>
    </Status>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdID>2</CmdID>
      <CmdRef>1</CmdRef>
      <Cmd>Sequence</Cmd>
      <!-- Sequence executed correctly -->
      <Data>200</Data>
    </Status>
    <Status>
      <MsgRef>1</MsgRef><CmdRef>2</CmdRef>
      <CmdID>3</CmdID>
      <Cmd>Alert</Cmd>
      <!-- OK, the user confirmed the action -->
      <Data>200</Data>
    </Status>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdRef>4</CmdRef>
      <CmdID>4</CmdID>
      <Cmd>Replace</Cmd>
      <TargetRef>./settings/wap_settings/CNN</TargetRef>
      <!-- OK, access point added -->
      <Data>200</Data>
    </Status>
    <Final/>
  </SyncBody>
</SyncML>
```

11.1.4 Package#4: Acknowledgement of DM Client status

This package is now empty as no actions are sent and client does not continue the protocol.

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
```

```
      <MsgID>2</MsgID>
      <Target>
        <LocURI>IMEI:493005100592800</LocURI>
      </Target>
      <Source>
        <LocURI>http://www.syncml.org/mgmt-server</LocURI>
      </Source>
    </SyncHdr>
    <SyncBody>
      <Status>
        <MsgRef>2</MsgRef>
        <CmdID>1</CmdID>
        <CmdRef>0</CmdRef>
        <Cmd>SyncHdr</Cmd>
        <Data>200</Data>
      </Status>
      <Final/>
    </SyncBody>
</SyncML>
```

11.2 Two-step protocol initiated by the DM Server

Operator initiates a regular antivirus software update on PDA clients. The DM Server checks the installed version in the first management section then updates the antivirus data in the second step.

11.2.1 Package#1: Initialization from DM Client to DM Server

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
    <Cred> <!-- Client credentials are optional -->
      <Meta>
        <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
        <Format xmlns='syncml:metinf'>b64</Format>
      </Meta>
      <Data><!-- base64 formatting of userid:password --></Data>
    </Cred>
    <Meta><!-- Maximum message size for the client -->
      <MaxMsgSize xmlns='syncml:metinf'>5000</MaxMsgSize>
    </Meta>
  </SyncHdr>
  <SyncBody>
    <Alert>
      <CmdID>1</CmdID>
      <Data>1200</Data> <!-- Server-initiated session -->
    </Alert>
    <Replace>
```

```
        <CmdID>3</CmdID>
        <Item>
          <Source><LocURI>./DevInfo/DevId</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>IMEI:493005100592800</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/Man</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>Device Factory, Inc.</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/Mod</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>SmartPhone2000</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/DmV</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>1.0.0.1</Data>
        </Item>
        <Item>
          <Source><LocURI>./DevInfo/Lang</LocURI></Source>
          <Meta>
            <Format xmlns='syncml:metinf'>chr</Format>
            <Type xmlns='syncml:metinf'>text/plain</Type>
          </Meta>
          <Data>US-en</Data>
        </Item>
      </Replace>
      <Final/>
    </SyncBody>
</SyncML>
```

11.2.2 Package#2: Initialization from DM Server to DM Client

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>IMEI:493005100592800</LocURI>
```

```
      </Target>
      <Source>
        <LocURI>http://www.syncml.org/mgmt-server</LocURI>
      </Source>
      <Cred> <!-- Server credentials -->
        <Meta>
          <Type xmlns='syncml:metinf'>syncml:auth-basic</Type>
          <Format xmlns='syncml:metinf'>b64</Format>
        </Meta>
        <Data><!-- base64 formatting of userid:password --></Data>
      </Cred>
    </SyncHdr>
    <SyncBody>
      <Status>
        <MsgRef>1</MsgRef><CmdRef>0</CmdRef>
        <Cmd>SyncHdr</Cmd>
        <CmdID>5</CmdID>
        <TargetRef>http://www.syncml.org/mgmt-server</TargetRef>
        <SourceRef>IMEI:493005100592800</SourceRef>
        <!-- Authenticated for the session -->
        <Data>212</Data>
      </Status>
      <Status>
        <MsgRef>1</MsgRef><CmdRef>1</CmdRef>
        <CmdID>6</CmdID>
        <Cmd>Alert</Cmd>
        <Data>200</Data><!-- OK -->
      </Status>
      <Status>
        <MsgRef>1</MsgRef><CmdRef>3</CmdRef>
        <CmdID>7</CmdID>
        <Cmd>Replace</Cmd>
        <Data>200</Data><!-- OK -->
      </Status>
      <Alert>
        <CmdID>2</CmdID>
        <Data>1100</Data> <!-- User displayable notification -->
        <Item></Item>
        <Item>
          <Data>Your antivirus software is being updated</Data>
        </Item>
      </Alert>
       <!-- Let's get the installed antivirus definition version number now -
->
      <Get>
        <CmdID>4</CmdID>
        <Item>
          <Target>
            <LocURI>./antivirus_data/version</LocURI>
          </Target>
        </Item>
      </Get>
      <Final/>
    </SyncBody>
</SyncML>
```

11.2.3 Package#3: DM Client response

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdID>1</CmdID>
      <Cmd>SyncHdr</Cmd>
      <Data>212</Data>
    </Status>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdRef>2</CmdRef>
      <CmdID>2</CmdID>
      <Cmd>Alert</Cmd>
      <Data>200</Data><!-- User notification OK -->
    </Status>
    <Status>
      <MsgRef>1</MsgRef>
      <CmdRef>4</CmdRef>
      <CmdID>4</CmdID>
      <Cmd>Get</Cmd>
      <TargetRef>./antivirus_data/version</TargetRef>
      <Data>200</Data><!-- Get OK -->
    </Status>
    <!-- Results for the Get: antivirus version number -->
    <Results>
      <MsgRef>1</MsgRef><CmdRef>4</CmdRef>
      <CmdID>3</CmdID>
      <Item>
        <Source>
          <LocURI>./antivirus_data/version</LocURI>
        </Source>
        <Data>antivirus-inc/20010522b/5</Data>
      </Item>
    </Results>
    <Final/>
  </SyncBody>
</SyncML>
```

11.2.4 Package#4: Continue with management operations

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
```

```
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>2</MsgID>
    <Target>
    <LocURI>IMEI:493005100592800</LocURI>
    </Target>
    <Source>
    <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <MsgRef>2</MsgRef>
      <CmdID>1</CmdID>
      <Cmd>SyncHdr</Cmd>
      <Data>212</Data>
    </Status>
    <!-- Send now antivirus updates -->
    <Replace>
      <CmdID>2</CmdID>
      <Meta>
        <Format xmlns='syncml:metinf'>b64</Format>
        <Type xmlns='syncml:metinf'>
          application/antivirus-inc.virusdef
        </Type>
      </Meta>
      <Item>
        <Target>
         <LocURI>./antivirus_data</LocURI>
        </Target>
        <Data><!-- Base64-coded antivirus file --></Data>
      </Item>
    </Replace>
    <Final/>
  </SyncBody>
</SyncML>
```

11.2.5 Restart protocol iteration with Package#3: DM Client response

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto> DM/1.3</VerProto>
    <SessionID>1</SessionID>
    <MsgID>3</MsgID>
    <Target>
      <LocURI>http://www.syncml.org/mgmt-server</LocURI>
    </Target>
    <Source>
      <LocURI>IMEI:493005100592800</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <MsgRef>2</MsgRef>
```

```
        <CmdID>1</CmdID>
        <Cmd>SyncHdr</Cmd>
        <Data>200</Data>
      </Status>
      <Status>
        <MsgRef>1</MsgRef>
        <CmdRef>2</CmdRef>
        <CmdID>2</CmdID>
        <Cmd>Replace</Cmd>
        <TargetRef>./antivirus_data</TargetRef>
        <!-- OK, antivirus update loaded -->
        <Data>200</Data>
      </Status>
      <Final/>
   </SyncBody>
</SyncML>
```

11.2.6 Package#4: Finish the protocol, no continuation

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
   <SyncHdr>
      <VerDTD>1.2</VerDTD>
      <VerProto> DM/1.3</VerProto>
      <SessionID>1</SessionID>
      <MsgID>3</MsgID>
      <Target>
        <LocURI>IMEI:493005100592800</LocURI>
      </Target>
      <Source>
        <LocURI>http://www.syncml.org/mgmt-server</LocURI>
      </Source>
   </SyncHdr>
   <SyncBody>
      <Status>
        <MsgRef>3</MsgRef>
        <CmdID>1</CmdID>
        <CmdRef>0</CmdRef>
        <Cmd>SyncHdr</Cmd>
        <Data>200</Data>
      </Status>
      <Final/>
   </SyncBody>
</SyncML>
```

12.Backward Compatibility

DM Servers MUST be compatible with all older minor and the same versions of the DM protocol.

DM Clients MUST be compatible with the same version of the DM protocol and SHOULD be compatible with the previous minor version of the DM Protocol and MAY be compatible with older versions of the DM protocol.

For example, a DM Server version 1.3 will have to be able to manage DM 1.1, DM 1.2 and DM 1.3 Clients. A DM Server version 1.2 will have to be able to manage DM 1.1 and DM 1.2 Clients. A DM 1.3 Client will have to be able to respond to DM 1.2 and DM 1.3 Servers.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
N/A	N/A	No prior 1.3 version

A.2 Draft/Candidate Version 1.3 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-DM_Protocol-V1_3	15 Oct 2008	All	Baseline to v1.3 using OMA-TS-DM_Protocol-V1_2_1-20080617-A .
	09 Sep 2009	8.7	Applied OMA-DM-DM13-2009-0038-CR_DMProto_Alert_Code_Enhancement.
	28 Oct 2009	All	Applied OMA-DM-DM13-2009-0080R01-CR_Backward_CompatibilitiyOMA-DM-DM13-2009-0094R02-CR_Package1_Description OMA-DM-DM13-2009-0012-CR_Alert__Non_Visual___Bug__Fix.
	02 Nov 2009	All	Removed CR_Alert_Non_Visual_Bug_Fix (was not Agreed). Applied OMA-DM-DM13-2009-0106R01-CR_Protocol_cleanup
	10 Dec 2009	All	Applied OMA-DM-DM13-2009-0118R06-CR_RelativeURI_Addressing.
	28 Dec 2009	All	Applied OMA-DM-DM13-2009-0121R02-CR_Protocol_Pkg1_DevInfoDetail OMA-DM-DM13-2009-0127-CR_VerProto_Bug_Fix_for_DM_Protocol
	29 Dec 2009	All	Fixed clerical errors, spelling mistakes, etc.
	03 Feb 2010	B.1.1	Applied OMA-DM-DM13-2010-0013R01-CR_Protocol_SCR OMA-DM-DM13-2010-0021R01-CR_RelativeURI_Enhancement OMA-DM-DM13-2010-0033-CR_SCR_Entries_for_Protocol OMA-DM-DM13-2009-0134R02-CR_Protocol_Bug_Fixes
	11 Feb 2010	All	Editorial clean-up of formatting
	15 Mar 2010	All	Changed all text to UK. Reapplied OMA-DM-DM13-2010-0021R01-CR_RelativeURI_Enhancement.
	23 Apr 2010	All	Editorial cleanup.
	26 Apr 2010	T.O.C	Update of T.O.C (figures)
	04 May 2010	3.1, 10.2, 10.2.5, 10.3.5, 11.2.1, 11.2.4	3.1, 10.2 and 10.3.5: grammatical correction Double quotes changed to single quotes
	05 May 2010	All	Formatting of bullets Formatting of note Double quotes changed to single quotes Snippets font changed from Courier to Courier New to harmonize the snippets fonts
Candidate Version OMA-TS-DM_Protocol-V1_3	25 May 2010	N/A	Status changed to Candidate by TP Ref # OMA-TP-2010-0221-INP_DM_V1.3_ERP_and_ETR_for_Candidate_approval
Draft Versions OMA-TS-DM_Protocol-V1_3	26 Aug 2010	7, 12	Applied OMA-DM-DM13-2010-0084-CR_Clarify_Large_Object_Usage OMA-DM-DM13-2010-0098R01-CR_Client_Backward_Compatibility
	20 Oct 2010	8	Re-applied change 3 of CR OMA-DM-DM13-2009-0121R02-CR_Protocol_Pkg1_DevInfoDetail

Document Identifier	Date	Sections	Description
	04 Nov	All	Applied: OMA-DM-DM13-2010-0114R01-CR_Protocol_TS_Cleanup
Candidate Version OMA-TS-DM_Protocol-V1_3	07 Dec 2010	N/A	Status changed to Candidate by TP Ref #OMA-TP-2010-0502-INP_DM_V1_3_ERP_and_ETR_for_Candidate_re_approval
Draft Versions OMA-TS-DM_Protocol-V1_3	15 Feb 2011	6 11 (new)	Applied: OMA-DM-DM13-2010-0130R01-CR_moving_DM20Protocol_to_DM13Protocol
	25 Mar 2011	8.3 8.8 (new)	Applied: OMA-DM-DM13-2011-0013R04-CR_Protocol_Tree_Change_Alert
	19 Apr 2011	8.3, 11	Applied: OMA-DM-DM13-2011-0017R05-CR_Protocol_Obj_Request OMA-DM-DM13-2011-0031R01-CR_MMA_Delegation_Intro
	10 May 2011	11 (deleted)	Applied: OMA-DM-DM13-2011-0039R03-CR_MMA_DM_Server_Protocol_TS
	04 Jul 2011	8.7.1.6	Applied: OMA-DM-DM13-2011-0050R01-CR_Generic_Alert_URN
	08 Jul 2011	8.7.1.6	Editorial changes
	14 Oct2011	5.2	Applied: OMA-DM-DM13-2011-0095-CR_URI_Clarificiation
	17 Oct2011	5	Update of the diagram as per CR OMA-DM-DM13-2011-0095-CR_URI_Clarificiation Editorial change from “relative “to” virtual”
	05 Dec2011	2.1, 5.2, 8.7.1, 8.9 (new)	Applied: OMA-DM-DM13-2011-0113R02-CR_DMPROTO_Update_for_Sessionless_Reporting
	23 Dec2011	3.2, 5	Applied: OMA-DM-DM13-2011-0099R04-CR_Virtual_URI_Mechanism
	13 Jan 2012	2, 3, 8, 8.3	Applied: OMA-DM-DM13-2011-0132R02-CR_CONR_Protocol
	31 Jan 2012	All	Applied: OMA-DM-DM13-2012-0012-CR_CONRR_Protocol_round_2 Restored cross-references in the the whole document Applied 2012 template to introduction section.
	09 Feb 2012	All	Applied: OMA-DM-DM13-2012-0053-CR_Node_addressing_removed_from_Proto
	14 Feb 2012	All	Applied: OMA-DM-DM13-2012-0044-CR_Protocol_Overview_Section_in_TS OMA-DM-DM13-2012-0046-CR_Requested_MO_in_DMPRO OMA-DM-DM13-2012-0051-CR_CONR_Protocol_editorial Update of cross-reference fields due to renumbering of sections
	16 Feb 2012	5.1, 5.8, 6, 6.7	Removed a typo and updated one cross-reference. Section 6 updated according to comment made against OMA-DM-DM13-2012-0046 in R&A DM13-12-010. Section 6.7 updated according to comment made against OMA-DM-DM13-2012-0051 in R&A DM13-12-010.
	22 Feb 2012	5.3	Applied OMA-DM-DM13-2012-0071R01-CR_Proto_CONRR_sessionID
	23 Feb 2012	4.1	Header removed by DSO according to Action Item DM-2012-A030
Candidate Version OMA-TS-DM_Protocol-V1_3	06 Mar 2012	N/A	Status changed to Candidate by TP Ref # OMA-TP-2012-0084-INP_DM_V1_3_ERP_and_ETR_for_Candidate_re_approval
Draft Versions OMA-TS-DM_Protocol-V1_3	21 Jun 2012	6.8.1 (new)	Applied OMA-DM-DM13-2012-0100R02-CR_Add_DM_Tree_Change_Alert_Msg_Struc

Document Identifier	Date	Sections	Description
	24 Aug 2012	B.1, B.2	Incorporated CRs: OMA-DM-DM13-2012-0107-CR_Change_SCR_References OMA-DM-DM13-2012-0108R03-CR_Check_and_Change_SCR_references
	26 Sep 2012	6, 10	Minor Editorial Changes to fix the word magic quotes in XML examples.
Candidate Version OMA-TS-DM_Protocol-V1_3	09 Oct 2012	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2012-0368R01-INP_DM_V1_3_ERP_and_ETR_for_Candidate_re_approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for DM Client

Item	Function	Reference	Requirement
DM-PRO-C-001-M	Support of Session Setup Phase	Section 6	
DM-PRO-C-002-M	Support of Session Abort	Section 6.1	
DM-PRO-C-003-O	Support of Multiple Messages	Section 7	
DM-PRO-C-004-O	Support of Large Object Handling. This is RECOMMENDED for clients.	Section 8	DM-PRO-LO-C-001-P AND DM-PRO-LO-C-002-P AND DM-PRO-LO-C-003-O AND DM-PRO-LO-C-004-O AND DM-PRO-LO-C-005-O AND DM-PRO-LO-C-006-O AND DM-PRO-LO-C-008-O AND DM-PRO-LO-C-009-O AND DM-PRO-LO-C-010-O
DM-PRO-C-005-M	Support of Management Phase	Section 6	
DM-PRO-C-006-M	Support for executing Management Commands	Section 5	
DM-PRO-C-007-O	Executing User Interaction Commands	Section 10	(DM-PRO-UI-C-001-O OR DM-PRO-UI-C-002-O OR DM-PRO-UI-C-003-O OR DM-PRO-UI-C-004-O OR DM-PRO-UI-C-005-O) AND DM-PRO-UI-C-006-O
DM-PRO-C-008-M	Support for sending Status and Result after receiving Management Operations	Section 6.5	
DM-PRO-C-009-M	Support for standard command Format and Status and Result reporting	Section 5	
DM-PRO-C-010-O	Support for sending asynchronous data via client initiated Alerts	Section 6.3	
DM-PRO-C-011-O	Sending Generic Alert	Section 6.7	DM-PRO-GAlert-C-001-O OR DM-PRO-GAlert-C-002-O

B.1.1 SCR for DM Session Setup Phase

Item	Function	Reference	Requirement
DM-PRO-Session-C-001-O	Support Server Notification	Section 5	
DM-PRO-Session-C-002-M	Sending Client Initiation and Device Info (Package #1) including Final element	Section 6.3	
DM-PRO-Session-C-	Sending Alert in the management	Section 6.3	

Item	Function	Reference	Requirement
003-O	session with data being either Client-Initiated 1201 or Server-Initiated 1200		
DM-PRO-Session-C-004-M	Sending Device Info in Replace Command in Package #1	Section 6.3	

B.1.2 SCR for Session Abort

Item	Function	Reference	Requirement
DM-PRO-Abort-C-001-O	Sending Session Abort Alert	Section 6.1	DM-PRO-Abort-C-004-O AND DM-PRO-Abort-C-005-O
DM-PRO-Abort-C-002-M	Receiving Session Abort Alert	Section 6.1	
DM-PRO-Abort-C-003-O	Session Abort message includes Status and Results of executed commands	Section 6.1.2	DM-PRO-Abort-C-001-O
DM-PRO-Abort-C-004-O	Include Final in Message	Section 6.1.2	
DM-PRO-Abort-C-005-O	Sender of Abort discards the response if response is received	Section 6.1.1	

B.1.3 SCR for Multiple Messages

Item	Function	Reference	Requirement
DM-PRO-Mul-C-001-M	Last message within multiple messages must contain Final	Section 7.2	
DM-PRO-Mul-C-002-M	If message that is not the last one within Multiple Messages then the Next Message or Abort Alert must be sent	Section 7.2	

B.1.4 SCR for Large Object

Item	Function	Reference	Requirement
DM-PRO-LO-C-001-O	Support for Large Object	Section 8	DM-PRO-LO-C-002-O AND DM-PRO-LO-C-003-O AND DM-PRO-LO-C-004-O AND DM-PRO-LO-C-005-O AND DM-PRO-LO-C-006-O AND DM-PRO-LO-C-008-O AND DM-PRO-LO-C-009-O AND DM-PRO-LO-C-010-O

Item	Function	Reference	Requirement
DM-PRO-LO-C-002-O	Respond with Status 213 when a data chunk that is not the last one is received	Section 8	
DM-PRO-LO-C-003-O	Management Commands inside Large Object are handled as Atomic	Section 8	
DM-PRO-LO-C-004-O	While sending data chunks all chunks except the last one must include “MoreData”	Section 8	
DM-PRO-LO-C-005-O	Data chunks must be sent in contiguous order without any new commands	Section 8	
DM-PRO-LO-C-006-O	Data that fits into a single message must be sent in a single message	Section 8	
DM-PRO-LO-C-007-O	Sending MaxObjSize to indicate size limitations for Package	Section 8	DM-PRO-C-004-O
DM-PRO-LO-C-008-O	Never create packages bigger than the size the Server indicated in MaxObjSize	Section 8	
DM-PRO-LO-C-009-O	Include Size in first data chunk	Section 8	
DM-PRO-LO-C-010-O	Compare actual size and the Size value and report if not equal	Section 8	
DM-PRO-LO-C-011-M	Indicate support for Large Object in DevDetail	Section 8	

B.1.5 SCR for User Interaction Commands

Item	Function	Reference	Requirement
DM-PRO-UI-C-001-O	Executing Display Alert	Section 10.2.1	DM-PRO-UI-C-006-O
DM-PRO-UI-C-002-O	Executing Confirm or Reject Alert	Section 10.2.2	DM-PRO-UI-C-006-O
DM-PRO-UI-C-003-O	Executing Text Input Alert	Section 10.2.3	DM-PRO-UI-C-006-O
DM-PRO-UI-C-004-O	Executing Single Choice Alert	Section 10.2.4	DM-PRO-UI-C-006-O
DM-PRO-UI-C-005-O	Executing Multiple Choice Alert	Section 10.2.4	DM-PRO-UI-C-006-O
DM-PRO-UI-C-006-O	Order of the Items MUST be used in the same order as in the DM message	Section 10.2	

B.1.6 SCR for Generic Alert

Item	Function	Reference	Requirement
DM-PRO-GAlert-C-001-O	The Generic Alert has a relation to a Management Object	Section 6.7	DM-PRO-GAlert-C-003-O
DM-PRO-GAlert-C-002-O	The Generic Alert does not have a relation to a Management Object	Section 6.7	
DM-PRO-GAlert-C-003-O	LocURI must reference the address to the corresponding Management Object	Section 6.7	
DM-PRO-GAlert-C-004-O	Support for Correlator	Section 6.7.1.10	DM-PRO-C-011-O
DM-PRO-GAlert-C-	Type must be included and it is	Section 6.7.1.6	DM-PRO-C-011-O

Item	Function	Reference	Requirement
005-O	RECOMMENDED to include URN or registered MIME-type as Type		
DM-PRO-GAlert-C-006-O	Support for importance level, Mark	Section 6.7.1.8	DM-PRO-C-011-O

B.2 SCR for DM Server

Item	Function	Reference	Requirement
DM-PRO-S-001-M	Support of Session Setup Phase	Section 6	
DM-PRO-S-002-M	Support of Session Abort	Section 6.1	
DM-PRO-S-003-M	Support of Multiple Messages	Section 7	
DM-PRO-S-004-M	Support of Large Object Handling	Section 8	
DM-PRO-S-005-M	Support of Management Phase	Section 6	
DM-PRO-S-006-M	Support for sending Management Commands	Section 5	
DM-PRO-S-007-O	Sending User Interaction Commands	Section 10	(DM-PRO-UI-S-001-O OR DM-PRO-UI-S-002-O OR DM-PRO-UI-S-003-O OR DM-PRO-UI-S-004-O OR DM-PRO-UI-S-005-O) AND DM-PRO-UI-S-006-O
DM-PRO-S-008-M	Support for sending Status and Results on Client Commands and Alerts	Section 6	
DM-PRO-S-009-M	Support of Generic Alert	Section 6.7	
DM-PRO-S-010-M	Support application layer authentication	Section 9	

B.2.1 SCR for DM Session Setup Phase

Item	Function	Reference	Requirement
DM-PRO-Session-S-001-O	Support Server Notification	Section 5	SCR-DM-NOTI-S-001-O
DM-PRO-Session-S-002-M	Support of receiving initiation message from client (Package #1), perform authentication and send initiation (Package #2)	Section 6.4	

B.2.2 SCR for Session Abort

Item	Function	Reference	Requirement
DM-PRO-Abort-S-001-O	Sending Session Abort Alert	Section 6.1	DM-PRO-Abort-S-004-O AND DM-PRO-Abort-S-005-O
DM-PRO-Abort-S-002-M	Receiving Session Abort Alert	Section 6.1	
DM-PRO-Abort-S-003-O	Session Abort message includes Status and Results of executed commands	Section 6.1.2	DM-PRO-Abort-S-001-O

Item	Function	Reference	Requirement
DM-PRO-Abort-S-004-O	Include Final in Message	Section 6.1.2	
DM-PRO-Abort-S-005-O	Sender of Abort must discard the response if response is received	Section 6.1.1	

B.2.3 SCR for Multiple Messages

Item	Function	Reference	Requirement
DM-PRO-Mul-S-001-M	Last message within multiple messages must contain Final	Section 7.2	
DM-PRO-Mul-S-002-M	If message that is not the last one within Multiple Messages then the Next Message or Abort Alert must be sent	Section 7.2	

B.2.4 SCR for Large Object

Item	Function	Reference	Requirement
DM-PRO-LO-S-001-M	Respond with Status 213 when a data chunk that is not the last one is received	Section 8	
DM-PRO-LO-S-002-M	Management Commands inside Large Object are handled as Atomic	Section 8	
DM-PRO-LO-S-003-M	While sending data chunks all chunks except the last one must include “MoreData”	Section 8	
DM-PRO-LO-S-004-M	Data chunks must be sent in contiguous order without any new commands	Section 8	
DM-PRO-LO-S-005-M	Data that fits into a single message must be sent in a single message	Section 8	
DM-PRO-LO-S-006-O	Sending MaxObjSize to indicate size limitations for Package	Section 8	
DM-PRO-LO-S-007-M	Never create packages bigger than the size the Client indicated in MaxObjSize	Section 8	
DM-PRO-LO-S-008-M	Include Size in first data chunk	Section 8	
DM-PRO-LO-S-009-M	Compare actual size and the Size value and report if not equal	Section 8	

B.2.5 SCR for User Interaction Commands

Item	Function	Reference	Requirement
DM-PRO-UI-S-001-O	Sending Display Alert	Section 10.2.1	DM-PRO-UI-S-006-O
DM-PRO-UI-S-002-O	Sending Confirm or Reject Alert	Section 10.2.2	DM-PRO-UI-S-006-O
DM-PRO-UI-S-003-O	Sending Text Input Alert	Section 10.2.3	DM-PRO-UI-S-006-O
DM-PRO-UI-S-004-O	Sending Single Choice Alert	Section 10.2.4	DM-PRO-UI-S-006-O
DM-PRO-UI-S-005-O	Sending Multiple Choice Alert	Section 10.2.4	DM-PRO-UI-S-006-O
DM-PRO-UI-S-006-O	Order of the Items MUST be	Section 10.2	

Item	Function	Reference	Requirement
	followed in the DM message		

B.2.6 SCR for Generic Alert

Item	Function	Reference	Requirement
DM-PRO-GAlert-S-001-M	Support for receiving, parsing and send Status back to client	Section 6.7	
DM-PRO-GAlert-S-002-O	Perform action from the data content in the Generic Alert	Section 6.7	

Appendix C. Protocol Values (Normative)

VerProto Codes	Description
DM/1.3	Indicates that this DM message uses the OMA Device Management Protocol defined by the Open Mobile Alliance.