Download Over the Air Specification

Approved Version 2.0.1 – 02 May 2011

Open Mobile Alliance

OMA-TS-DLOTA-V2_0_1-20110502-A

Contents

1. SCOPE 5
2. REFERENCES 6
   2.1 NORMATIVE REFERENCES 6
   2.2 INFORMATIVE REFERENCES 7
3. TERMINOLOGY AND CONVENTIONS 9
   3.1 CONVENTIONS 9
   3.2 DEFINITIONS 9
   3.3 ABBREVIATIONS 11
4. INTRODUCTION 13
5. OMA DOWNLOAD PROCESS 14
   5.1 MEDIA OBJECT DISCOVERY PROCESS 15
      5.1.1 Step 1; The Download Descriptor is transferred to the device 15
   5.2 MEDIA OBJECT DOWNLOAD PROCESS 16
      5.2.1 Step 2; The Downloading Agent is launched, the Download Descriptor is processed 16
      5.2.2 Step 3; Capabilities Check 17
      5.2.3 Step 4; User Confirmation 18
      5.2.4 Step 5a; Media Object retrieval 19
      5.2.5 Step 5b; License Retrieval 25
      5.2.6 Step 6; Sending Download Completion Notification 26
   5.3 OBJECT INSTALLATION PROCESS 26
      5.3.1 Step 7; Installation 27
      5.3.2 Step 8; Downloading of Post Licenses 29
      5.3.3 Step 9; Sending Installation Notification 29
   5.4 AFTER THE OBJECT INSTALLATION PROCESS 31
      5.4.1 Step 10; Download Confirmation and Next Step 31
      5.4.2 Step 11; Sending Deletion Notification 32
      5.4.3 Local Content Presentation 32
      5.4.4 Persistence of Download Descriptor elements 32
   5.5 HTTP SPECIFIC FUNCTIONALITY 33
      5.5.1 Client capability advertisement 33
      5.5.2 State Management of download transaction 35
      5.5.3 Transparency of Download Descriptor mechanism 36
   5.6 SERVER INITIATED AUTOMATIC DOWNLOAD 36
6. STATUS REPORT FUNCTIONALITY 38
   6.1 DLOTAV2 STATUS REPORTS 38
   6.2 STATUS REPORT FORMATTING 41
7. DOWNLOAD DESCRIPTOR 43
   7.1 DOWNLOAD DESCRIPTOR 43
   7.2 DOWNLOAD DESCRIPTOR ELEMENTS AND ATTRIBUTES 43
      7.2.1.1 DD 44
      7.2.2 vendor 44
      7.2.3 product 45
      7.2.4 updatedDDURI 56
      7.2.5 reservation 56
      7.2.6 nextURL 59
   7.3 EXTENSIBILITY 59
      7.3.1 Media type with custom installation commands 59
   7.4 XML SYNTAX FOR DOWNLOAD DESCRIPTOR 59
8. DLOTAV2 SECURITY 61
   8.1 DOWNLOAD AGENT AUTHENTICATION 61
   8.2 SERVER AUTHENTICATION 61

8.3 SERVER AUTHORIZATION 61
8.4 CONFIDENTIALITY AND INTEGRITY PROTECTION 61
9. DOWNLOAD OTA OVER BROADCAST BEARER (INFORMATIVE) 63
10. RELATIONSHIP TO JAVA™ MIDP OTA 64
11. BACKWARDS COMPATIBILITY 68
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 69
A.1 APPROVED VERSION HISTORY 69
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 70
B.1 CLIENT CONFORMANCE REQUIREMENTS 70
B.2 SERVER CONFORMANCE REQUIREMENTS 72
APPENDIX C. DIFFERENCE BETWEEN COMPOUND PRODUCT AND PRODUCT 75
APPENDIX D. EXAMPLE OF DOWNLOAD DESCRIPTOR (INFORMATIVE) 77
D.1 EXAMPLE OF BASIC DOWNLOAD DESCRIPTOR 77
D.2 EXAMPLE OF BASIC DOWNLOAD DESCRIPTOR WITH INSTALLNOTIFYURI 77
D.3 EXAMPLE FOR MULTIPLE SERVER SUPPORT 77
D.4 EXAMPLE OF ENVIRONMENT ELEMENT 77
D.5 EXAMPLE OF ENVIRONMENT ELEMENT USE FOR WRAPPING A MIDP JAD 78
D.6 EXAMPLE WITH MULTIPLE PRODUCTS AND MEDIA OBJECTS 79
D.7 EXAMPLE WITH LICENSE ELEMENTS 80
D.8 EXAMPLE WITH ADDITIONAL TEXTUAL METADATA 80
D.9 EXAMPLE WITH LICENSE ELEMENT FOR ACQUIRING RIGHTS ONLY (SUPERDISTRIBUTION). 81
APPENDIX E. EXAMPLE OF DOWNLOAD TRANSACTION (INFORMATIVE) 83
E.1 HTTP REQUEST TO VIEW A DOWNLOAD SERVICE PAGE 83
E.2 HTTP REQUEST FOR DOWNLOAD DESCRIPTOR 83
E.3 HTTP REQUEST TO INSTALL A MEDIA OBJECT 84
E.4 INSTALL STATUS VIA HTTP POST REQUEST 84
E.5 PAUSE AND RESUME MEDIA RETRIEVAL 84
APPENDIX F. MEDIA TYPE REGISTRATION 86

Figures

Figure 1: OMA Download Process. 15
Figure 2: Installation Process 28
Figure 3: Compound Product 76
Figure 4: Product 76

Tables

Table 1: Status codes used for Reservation Related Notifications 38
Table 2: Status codes used for Download Completion Notification 39
Table 3: Status codes used for Installation Notification 40
Table 4: Status codes used for Deletion Notification 41
Table 5: Difference between Compound Product and Product 75

1. Scope

The OMA Download Over-the-Air (OTA) specifications define procedures for enabling the downloading of Media Objects hosted on a server to a client. This specification defines the Download OTA version 2.0 technical specification.

OMA Download OTA version 1.0 provides a mechanism for user-initiated download of content, such as ring-tones, images, and applications. While OMA Download OTA provides much of the functionality needed to provide for a more reliable download solution than basic HTTP, other protocols exist in the mobile industries that provide functionality beyond that of OMA Download OTA version 1.0.

Download OTA version 2.0 is an evolution of Download OTA version 1.0. The purpose of Download OTA version 2.0 is to add functionality that was missing from version 1.0. Backward compatibility can be achieved by also adding support for version 1.0 Download Descriptors to the Download User Agent.

2. References

2.1 Normative References

[DDXSD] "Download Descriptor Schema", Open Mobile Alliance™, OMA-SUP-XSD_dd-V2_0. URL: http://www.openmobilealliance.org/

[DLOTAv1] "OMA Download", Version 1.0, Open Mobile Alliance™, OMA-DL-V1_0. URL: http://www.openmobilealliance.org/

[GAA] "Generic Authentication Function; Access to Network Application Functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS); (Release 6)", 3GPP TS 33.222 v6.4.0. URL: http://www.3gpp.org/ftp/Specs/html-info/33222.htm

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.3, Open Mobile Alliance™, OMA-ORG-IOP_Process-V1_3. URL: http://www.openmobilealliance.org/

[JADXSD] "XML Schema for JAD MIDP Extension", Open Mobile Alliance™, OMA-SUP-XSD_dd_midpjad-V2_0. URL: http://www.openmobilealliance.org/

[MIDP] "Mobile Information Device Profile", Version 1.0, 2000, SUN Microsystems, URL: http://java.sun.com/

[MIDPOTA] "Over The Air User Initiated Provisioning Recommended Practice", Version 1.0, 2001, SUN Microsystems. URL: http://java.sun.com

[MIDP20] "Mobile Information Device Profile 2.0 (JSR-118)", URL: http://www.jcp.org/jsr/detail/118.jsp

[OMADM] "OMA Device Management", Version 1.2, Open Mobile Alliance™, OMA-ERP-DM-V1_2. URL: http://www.openmobilealliance.org/

[OMADRMv1] "Digital Rights Management", Version 1.0, Open Mobile Alliance™, OMA-DRM-V1_0. URL: http://www.openmobilealliance.org/

[OMADRMv2] "Digital Rights Management", Version 2.0, Open Mobile Alliance™, OMA-ERP-DRM-V2_0. URL: http://www.openmobilealliance.org/

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997. URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC2246] T. Dierks and C. Allen, "Transport Layer Security (TLS) Version 1.0", IETF RFC 2246, Jan 1999. URL: http://www.ietf.org/rfc/rfc2246.txt

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC 2616, June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", IETF RFC 2617, June 1999. URL: http://www.ietf.org/rfc/rfc2617.txt

[RFC2818] E. Rescorla, "HTTP over TLS", IETF RFC 2818, May 2000. URL: http://www.ietf.org/rfc/rfc2818.txt

[RFC3268] P. Chown, "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", IETF RFC 3268, June 2002. URL: http://www.ietf.org/rfc/rfc3268.txt

[RFC3986] T. Berners-Lee, R. Fielding and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 3986, August 1998. URL: http://www.ietf.org/rfc/rfc3986.txt

[RFC4234] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", IETF RFC 4234, November 1997. URL: http://www.ietf.org/rfc/rfc4234.txt

[SI] "WAP Service Loading Specification", WAP Forum™, WAP-167-ServiceInd-20010731-. URL: http://www.openmobilealliance.org/

[SL] "WAP Service Indication Specification", WAP Forum™, WAP-168-ServiceLoad-20010731-. URL: http://www.openmobilealliance.org/

[UAProf] "OMA User Agent Profile", Version 2.0, Open Mobile Alliance™, OMA-ERP-UAProf-V2_0. URL: http://www.openmobilealliance.org/

[WAPCert] "WAP Certificate profile Specification", WAP Forum™, WAP-211-WAPCert-20010522-a. URL: http://www.openmobilealliance.org/

[WAPPush] "Push Message Specification", WAP Forum™, WAP-251-PushMessage-20010322-a.
"Push Proxy Gateway Service Specification", WAP Forum™, WAP-249-PPGService-20010713-a.
"Push OTA Protocol Specification", WAP Forum™, WAP-235-PushOTA-20010425-a.
URL: http://www.openmobilealliance.org/

[WAPTLS] "WAP TLS Profile and Tunneling Specification", WAP Forum™, WAP-219-TLS-20010411-a. URL: http://www.openmobilealliance.org/

[WSP] "Wireless Session Protocol Specification", WAP Forum™, WAP-230-WSP-20010705-a. URL: http://www.openmobilealliance.org/

[WTLS] "Wireless Transport Layer Security Specification", WAP Forum™, WAP-261-WTLS-20010406-a. URL: http://www.openmobilealliance.org

[XMLSchema] D. Beech, M. Maloney, and N. Mendelsohn, "XML Schema Part 1: Structures" W3C Recommendation, May 2001. URL: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
P. Biron and A. Malhotra, "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001. URL: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

2.2 Informative References

[BCMCS] "Broadcast/Multicast Services - Stage 1", 3GPP2 S.R0030-A v1.0 – Revision A. URL: http://www.3gpp2.org/

[DLREQ] "Download Over the Air Requirements", Version 2.0, Open Mobile Alliance™, OMA-RD-DLOTA-V2_0. URL: http://www.openmobilealliance.org/

[DLARCH] "Download Over the Air Architecture", Version 2.0, Open Mobile Alliance™, OMA-AD-DLOTA-V2_0. URL: http://www.openmobilealliance.org/

[DVB-H] "Digital Video Broadcasting (DVB): Transmission System for Handheld Terminals (DVB-H)", ETSI EN 302 304. URL: http://portal.etsi.org/

[HTTPSM] "HTTP State Management Specification", WAP Forum™, WAP-223-HTTPSM-20001213-a. URL: http://www.openmobilealliance.org/

[MIDPIMPBP] Implementation Best Practices for OMA DRM v1.0 protected MIDlets, Version 1.0, May 5th, 2004, available from http://www.forum.nokia.com/info/sw.nokia.com/id/a307e7ea-db42-471e-8e2e-1dd73b8e841a/Implementation_Best_Practices_For_OMA_DRM_v1_0_Protected_MIDlets_v1_0_en.pdf.html

[MBMS] "Multimedia Broadcast/Multicast Service (MBMS)", 3GPP TS 22.146 – Release 6 Services and Systems Aspects.
"Multimedia Broadcast/Multicast Service (MBMS) user services", 3GPP TS 22.246 – Release 6; Services and Systems Aspects.
URL: http://www.3gpp.org/

[MMS] "OMA Multimedia Messaging Service", Version 1.3, Open Mobile Alliance™, OMA-ERP-MMS-V1_3. URL: http://www.openmobilealliance.org/

[OMNA] "Open Mobile Naming Authority", URL: http://www.openmobilealliance.org/tech/omna/

[RFC2045] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", IETF RFC 2045, November 1996. URL: http://www.ietf.org/rfc/rfc2045.txt

[RFC2046] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", IETF RFC 2046, November 1996. URL: http://www.ietf.org/rfc/rfc2046.txt

[XML] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000. URL: http://www.w3.org/TR/2000/REC-xml-20001006

[XMLNS] Tim Bray, Dave Hollander and Andrew Layman, "Namespaces in XML", W3C Recommendation, January 1999. URL: http://www.w3.org/TR/1999/REC-xml-names-19990114/

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.

3.2 Definitions

Compound Product	A Compound Product is Product where all Licenses and Media Objects must be Downloaded and installed as one entity. If one of the License or one of the Media Objects fail to be downloaded or installed, the complete Compound Product SHALL be discarded by the Download Agent. Once the Compound Product is installed, each Media Object in a Product is handled as an independent Media Object.
Content Delivery	The actual delivery of the Media Object, for example by means of a HTTP GET, to the client device.
Content Download	The whole transaction including discovery, delivery of content and confirmation of download.
Content Handler	An entity in the mobile device responsible for the processing of a particular media type. The content handler typically handles issues related to installation of content, in addition to execution of content. The actual processing of retrieved content is outside the scope of this specification.
Content Storage	The physical location of the Media Object to be downloaded.
Deletion Notification	The Deletion Completion Notification indicates to the Status Report Server that the Media Object or Product has been removed from the device, and the Media Object or Product is no longer available to the user.
Discovery Application	A user agent in the device that discovers media on behalf of the user. The End User discovers content on the Web by using a Web browser or an application specifically created for a type of content. A picture editor may discover pictures, a melody composer may discover melodies, and an application manager may discover applications (e.g. games) on dedicated Web sites. Email and MMS messages may contain Web addresses to Media Objects available for downloading. These types of applications are collectively referred to as a *Discovery Application.*
Discovery Process	The process by which the user or device finds a resource (i.e. a Media Object) that he wants to load onto his device. The discovery can take place for example by means of a browser, a dedicated discovery application, a received message, or some offline means (like a newspaper).
Download Agent	An abbreviated form of Download User Agent.
Download Completion Notification	The Download Completion Notification indicates to the Status Report Server that the Download Agent successfully downloaded the Media Object or Product. At the point where this notification is sent the Media Object or Product is not available to the user.
Download Descriptor	Metadata about *Media Objects, Products* and instructions to the *download agent* for how to download these. The object triggers the Download Agent in the client device. It describes the Media Objects to be downloaded and allows the client device to decide if it has the capabilities to install and render/execute the Media Objects. Media Objects are grouped within Products to create an association between Media Objects.

Download Protocol	The actual delivery of an object is performed using the protocol specified in the Download Descriptor. The only mandatory protocol as defined in this specification is [RFC2616] (or [WSP] if the environment is WAP 1.x).
Download Server	A Web server hosting *Media Objects* available for download. It is responsible for the download transaction from the server perspective. It handles download session management including actions triggered by the installation status report.
Download Service	The overall service that a client device is exposed to when it wants to select a Product or Products and execute a download of it. A download service is typically constructed with the help of the abstract building blocks Presentation Server, Download Server and Content Storage.
Download Size	The number of bytes returned as a result of a successful request to the objectURI.
Download User Agent	A user agent in the device responsible for downloading *Media Objects* described by a *Download Descriptor*. It is responsible for the download transaction from the client perspective. It is triggered by the reception or activation of a Download Descriptor.
Generic Content	The concept of Generic Content includes any MIME media type except the Java™ JAR media type. For this media type please see [MIDPOTA].
Installation Notification	A Status Report message from the client to the server. It indicates to the server that the Download Agent has successfully installed the Media Object or Product, and that the content (to the best knowledge of the Download Agent) will be made available to the user.
License	An electronic certificate that enables the use of the Media Object. This could for instance be an OMA DRM v2 Rights Object. See [OMADRMv2].
License Agent	The entity in the device that manages License for Media Objects on the device.
Media Object	A resource on a Web server that can be downloaded. It may be a single object (often referred to as a file), or a container consisting of multiple objects. The mechanism for the latter may be MIME-multipart. There are no restrictions as to the characteristics of the Media Object, but the transfer encoding has to make it compatible with an HTTP (or WSP) transport. The download of a Media Object is the ultimate goal of each transaction undertaken with the protocol defined in this specification.
Media Object Installer	The Media Object Installer is responsible for the preparation for and execution of the installation of a particular Media Object. The Installer is often implemented as part of the Content Handler of the particular media type or as part of a file system manager.
Media Type	A MIME media type [RFC2046].
Memory Space	A device may support several memory spaces for example flash memory and removable media.
MIDP OTA Provisioning	The JAVA™ MIDP OTA Provisioning is defined in [MIDPOTA].
Presentation Server	A Web server presenting a download service to the user. It is one of the possible discovery mechanisms. The client device may browse a Web or WAP page at the presentation server and be redirected to the Download Server for the OMA Download transaction.
Product	A Product is a logical grouping of one or more independent Media Objects and optional associated Licenses that are downloaded and installed as one logical group. This can be an arbitrary set defined by a content provider. Once the product is installed, each Media Object in a Product is handled as an independent Media Object.
Server	All Servers in this specification are abstract, i.e. logical, entities. They are used in the specification only to help the reader to separate between different functional elements that may be implemented and deployed in any configuration.
Status Report	A message sent from the mobile device to a server to indicate the positive or negative outcome of a download transaction. In the context of Content Download the Status Report terminates the "download session" (or "download transaction").

Status Report Server	A WEB server accepting status reports from the download agent.
Unallocated Memory	Memory that is "free" i.e. available for writing to.
Well-intentioned attempt	A "well-intentioned attempt to send an Installation Status Report" means that the client device sends a Status Report under circumstances where the network connection is known (to the extent possible) to be present, and the Status Report is known to be properly formatted. If there is no network connection then an attempt to send a request should not be regarded as well-intentioned.

3.3 Abbreviations

3GPP	3rd Generation Partnership Project
3GPP2	3rd Generation Partnership Project 2
AMS	Application Management Software as defined in [MIDP20]
BCMCS	Broadcast Multicast Services
CID	Content Identifier
DCF	DRM Content Format as specified in [OMADRMv1] and [OMA DRMv2]
DD	Download Descriptor
DRM	Digital Rights Management
DVB-H	Digital Video Broadcasting – for Handheld terminals
DLOTA	Download Over the Air
GAA	General Authentication Architecture
HTTP	HyperText Transfer Protocol
JAD	Java™ Application Descriptor
JAR	Java™ Archive
J2ME	Java™ 2 Micro Edition
MBMS	Multimedia Broadcast Multicast Services
MIME	Multipurpose Internet Mail Extensions
MIDP	Mobile Information Device Profile
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
OMNA	Open Mobile Naming Authority
OTA	Over The Air
SI	Service Indication
SL	Service Loading
TLS	Transport Layer Security
RP	Recommended Practices
UAProf	User Agent Profile
URL	Universal Resource Locator
URI	Universal Resource Identifier
WAP	Wireless Application Protocol
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security

XML	Extensible Markup Language

4. Introduction

OMA Download Over-the-Air (DLOTA) provides a flexible mechanism for downloading Media Objects of any type and size from a network. A typical Media Object targeted by OMA DLOTA is downloaded and stored in the device, for example, in order to personalise the device or enhance its functionality. Examples of such Media Objects are ring-tones, background images, music/video files and applications. OMA DLOTA version 2.0 is also agonistic of the protection mechanism, i.e. OMA DRM [DRM2.0] or others, used to protect the Media Object.

Details of use cases and the requirements are specified in the OMA DLOTA version 2.0 requirements document [DLREQ]. The DLOTA architecture is specified in OMA DLOTA version 2.0 architecture document [DLARCH].

5. OMA DOWNLOAD Process

During the download and installation process the user SHOULD be given the opportunity to control the download and to determine object specific terms. For any operation, the user SHOULD be informed of progress and given an opportunity to cancel the activity. The user interface of the device SHOULD allow the user to abort the download operation at appropriate points during the download and installation process (i.e. before a well-intentioned attempt to send an installation notification has been done). When multiple Media Objects and associated licenses are downloaded, DLOTAv2 provides the device with the necessary information to allow it to display a single progress bar/indicator so that from a user perspective the download of Media Object(s) and associated license(s) is viewed as a single process. When the Product is a Compound Product it is RECOMMENDED that a single progress bar/indicator is used to provide feedback to the user.

When a Media Object or Product is installed, and if an Installation Notification has been requested in the Download Descriptor, a confirmation MUST be sent to the server to indicate that the installation has completed. If the Download Descriptor does not include a request for an Installation Notification then no such confirmation will be sent.

If an `installNotifyURI` has been defined in the Download Descriptor, then errors during the download process MUST be reported using the status report mechanism. The server may use the status report, communicating both success and failure of the transaction, for accounting or for other customer service needs.

A Download Descriptor MAY contain multiple Media Objects grouped as a Product. It MAY also contain multiple Products. An `installNotifyURI` can be defined either for one or more Media Objects of a Product, or for the Product as a whole. If defined for a Product as a whole, any notification MUST be sent after the Download Agent has handled all Media Objects of the Product. Otherwise, if the URI is defined for a specific Media Object, any notification MUST be send as soon as the Download Agent has handled that specific Media Object. If `installNotifyURI` is defined for a Product, `installNotifyURI` MUST NOT be specified for separate Media Objects of corresponding Product. The Download Agent MUST discard any `installNotifyURI` request for separate Media Objects if `installNotifyURI` is defined for a product.

If a Product is a Compound Product, the `installNotifyURI` MUST NOT be specified for separate Media Objects of the Compound Product. It MAY only be specified for the Product as a whole.

A Download Descriptor MAY contain the license element, when present this element identifies the DRM Agent to which the contents of the license element should be provided. This allows a Download Service to indicate to a Download Agent that the content is DRM protected and a license should be retrieved by the appropriate DRM Agent in order to allow consumption of the content.

It is RECOMMENDED that devices support the capability to download multiple Media Objects grouped within a single Download Descriptor. If a device supports the download of multiple Media Objects grouped within a single Download Descriptor it MUST support at least 1 Product consisting of at least 10 Media Objects. If the device supports multiple Products then it MUST support at least 10 Media Objects per Product. If a device supports the download of multiple Media Objects grouped within a single Download Descriptor it is RECOMMENDED that the device supports UAProf to advertise the number of Products and Media Objects per Product that it supports. See section 5.3.2 for details on sending Installation Notifications.

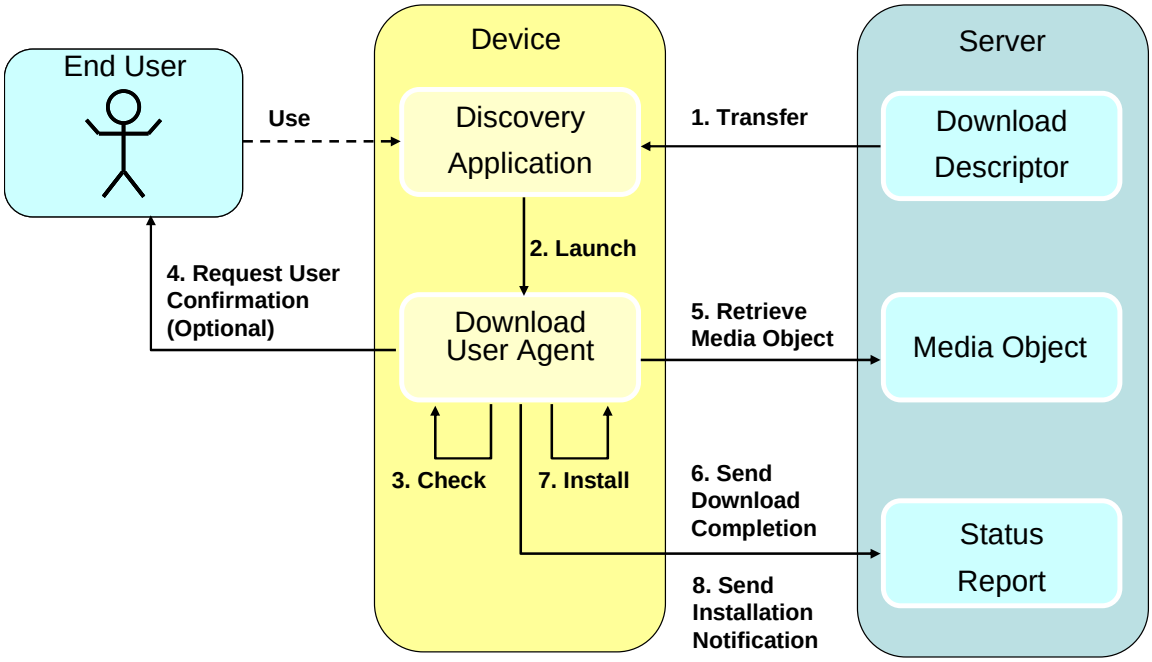

Figure 1: OMA Download Process.

5.1 Media Object Discovery Process

While using the Discovery Application, the user is typically presented with a reference to the Download Descriptor. The reference may be on a Web page, or inside an email or MMS message, or stored in memory or in an accessory attached to the phone.

5.1.1 Step 1; The Download Descriptor is transferred to the device

The transfer protocol used depends on the where the Download Descriptor is located and on the requirements of the transfer.

The device hosting the Download Agent and the Download Server MUST at a minimum support HTTP [RFC2616] and TLS [RFC2246]. The Download Agent and the Download Server MAY support WSP [WSP], as well as other protocols. If the Download Agent supports WSP then it MUST also support WTLS [WTLS]. The Download Descriptor may be retrieved from the Download Server, or the presentation server, depending on requirements of the deployment.

The device hosting the Download Agent MAY also support reception of the Download Descriptor using a mechanism such as MMS, email or some instant messaging protocol.

The Content-Type parameter of the transport protocol or message container (or equivalent) MUST be used to detect the Download Descriptor media type; a charset parameter MAY also be used to indicate the character set of the Download Descriptor.

It is recommended that servers do not put multiple Download Descriptors into one and the same transport entity (e.g. multipart/mixed). This would make transaction management, and interpretation of the multipart semantics, unnecessarily complicated. The device MAY support this case, but is not required to.

For details on state management during the transaction see Section 5.5.2 State Management of Download Transactions.

For details related to security see Section 8 DLOTAv2 Security.

5.1.1.1 Co-Delivery of Download Descriptor and Media Objects

When the Download Descriptor is delivered to the device the server system may also deliver one or more of the Media Objects specified in the Download Descriptor in a single delivery operation. This is described as a "Content Download transaction with co-delivery of Descriptor and Media Objects". If the device in the capability negotiation indicates that it supports multipart/related ([RFC2387]) or application/vnd.wap.multipart.related ([WSP]) then it MUST be able to process such a multipart with the Download Descriptor as the first part and the Media Objects as the second and further parts. The CID (Content ID) mechanism MUST be used in the Download Descriptor to reference the Media Object in a related multipart.

When one or more Media Objects are Co-Delivered with the Download Descriptor the process continues as described in Section 5.2 Object Download Process, with the exception that the Co-Delivered Media Objects are retrieved locally (in step 5) rather than from an external location. The same steps of "Launch of Download Agent (Step2)", "Capabilities check (Step 3)" and "User Confirmation (Step 4)" are still performed.

If the download transaction is aborted before "Installation (step 7)" (or optionally during "Sending Installation Notification (step 7)") then the Media Object MUST be discarded from the device.

5.2 Media Object Download process

Object Download Process is the process by which Media Objects are downloaded onto the device and made ready for installation.

To download Media Objects, the Download Agent that is responsible for the processing of the Download Descriptor performs a number of actions. If the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT provide the user with appropriate feedback, however for all other cases the Download Agent SHOULD provide the user with appropriate feedback if one of the actions fails. In all cases the Download Agent MUST use the Status Report mechanism (if requested in the Download Descriptor) to give the server infrastructure feedback about a possible failure of the download event.

Within OMA Download OTA version 2.0, there are four types of notifications. These notifications are used to provide feedback on the status of the download and installation operations. These notifications can be used by a service to determine how many Media Objects were downloaded and installed, or if a large number of users aborted the download when they were presented with a formal description of the Media Object. For details related to the Status Report mechanism see section 6 Status Report Functionality.

With OMA Download OTA version 2.0, it is possible to download complex structures of Media Objects and associated Digital Rights. In addition it is also possible to define when a Download Transaction shall be started and Download Transaction can be automatically initiated from a server. It is envisioned that many of these tasks will be executed in the background. The user must be able to view the status of old and ongoing downloads. The Download Agent MUST be able to keep a status log of the 5 latest successful and failed download transactions that can rendered by the user.

5.2.1 Step 2; The Downloading Agent is launched, the Download Descriptor is processed

The Download Agent is launched.

The Download Descriptor MUST be processed according to the rules defined in Section 5.2.1.1 Processing Rules. If the Download Agent encounters an error when parsing and interpreting the Download Descriptor (e.g. a syntax error) and the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT display an error message to the user. For all other cases the Download Agent SHOULD display an appropriate error message to the user. If an `installNotifyURI` element is present in the Download Descriptor then the Download Agent MUST post an "Invalid Descriptor" status report. The device MAY skip sending a status report if it is unable to parse the Download Descriptor. The device MAY select any of the defined `installNotifyURI`, in case the Download Descriptor contains multiple Products and or Media Objects.

If the DDVersion attribute of the Download Descriptor indicates a "major" version that is not supported by the client device it SHOULD send an "Invalid DDVersion" status code to the installNotifyURI and reject the Download Descriptor.

If an error occurs the scenario stops at this point.

5.2.1.1 Processing Rules

This specification strives to ensure forward compatibility. The parsing and processing of first generation device implementations should not become incompatible as additional elements are deployed. Thus, when processing the Download Descriptor, the following rules apply:

- Optional elements, as defined in this specification, MUST be ignored if not supported.
- Unknown elements MUST be ignored (these elements MAY be defined outside this specification).
- If an element occurs more than once, all but the first occurrence MUST be ignored.
  - The exception to this rule is the elements that can occur multiple times (e.g. type, mediaObject, product, server, text, etc). If there are more occurrences of an element than the Download Agent can support then the Download Agent SHOULD accept as many elements as it supports and after that ignore all subsequent element values as redundant (i.e. in case of multiple occurrences of an element the first one has the highest priority). For more information on the elements that can be occurred multiple times, see section 7.4 XML Syntax for Download Descriptor.

5.2.2 Step 3; Capabilities Check

The Download Agent MUST use the information in the Download Descriptor, at a minimum the size, installSize (if present) and type elements, to check whether the device is capable of using and/or rendering all the Media Objects included in the Download Descriptor. This is to prevent the download of Media Objects that can not be consumed on the device.

A device MUST use the size element to determine if it has sufficient Unallocated Memory to download all the Media Objects included in the Download Descriptor. If the installSize element is present in the Download Descriptor the device MUST use this element to determine if it has sufficient Unallocated Memory to download and install all the Media Objects. If the installSize element is not present in the Download Descriptor the device MUST use the size element to determine if it has sufficient Unallocated Memory to download and install all the Media Objects. If the device does not have sufficient memory to download and install all the Media Objects and the Download Descriptor contains the suppressUserConfirmation element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST terminate the download session and post a 901 "Insufficient Memory" status report. For all other cases the Download Agent MUST aid the user in reviewing memory usage and freeing sufficient memory to enable the device to download and install the new Media Objects. As a result of this event there are 3 possible outcomes:

(i) There is sufficient memory to download and install the Media Objects. In this case the device should continue to download and install the Media Objects.

(ii) There is sufficient memory to download the Media Object but insufficient memory to install the Media Objects. In this case the device SHOULD inform the user that the Media Objects can not be installed until further memory is made available or if applicable a different memory space (with sufficient memory) is selected and give the user the option to continue with the Download only.

(iii) There is insufficient memory to download the Media Object. In this case the Download Agent MUST post an "Insufficient Memory" status report and notify the end user that it was not possible to download and install the Media Object.

If the Download Descriptor contains multiple Media Objects or Products and, after freeing some memory, there is still insufficient memory to store all the Media Objects on the device, but enough memory to store some of the Media Objects, the Download Agent SHOULD give the end user a chance to select which of the Media Objects or Products that shall be downloaded.

If a Product is a Compound Product, the Download Agent MUST handle the Product as if it is a single Media Object when performing any capability check. The Media Objects of a Compound Product SHOULD NOT be installed when any of the Media Objects of the Compound Product does not meet the capabilities of the device. The Download Agent MUST show the Compound Product as a single selectable item when giving the end user a chance to select Media Objects in case of insufficient memory.

If the based on the elements in the Download Descriptor the Download Agent concludes that the device lacks the capability to perform a successful installation of a Media Object or a Compound Product, for any reason other than insufficient memory, the Download Agent MUST post a "Device Aborted" status report. If the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to 'Always' and the server can be authorized (see section 8.3) then the Download Agent MUST NOT notify the end user. In all other cases the Download Agent SHOULD notify the user.

If there is more than one `type` element in the Download Descriptor then the device MAY continue with the download transaction even if not all media types defined in the `type` element are supported by the device.

5.2.3 Step 4; User Confirmation

User confirmation is an important part of the download and installation process, and is required to prevent 3rd parties from surreptitiously installing Media Objects on the device without the users knowledge, however, there are use cases where it is desirable to suppress the user confirmation, for example: services where the Media Objects will be downloaded during the night when the user will not be available.

In order to enable these use cases the Download Server can signal that user confirmation should not be requested through inclusion of the `suppressUserConfirmation` element in the Download Descriptor. The Download Agent will only honour this request if the Download Server is authorized. See section 8.3 Server Authorization for details of how the Download Agent can determine if the Download Server is authorized to request suppression of the user confirmation.

If the Download Descriptor includes the `suppressUserConfirmation` element and the value is equal to 'Always' and the Download Server can be authorized then the Download Agent MUST NOT request user confirmation.

If the Download Descriptor includes the `suppressUserConfirmation` element and the value is equal to 'UserConfirmStepOnly' and the Download Server can be authorized then the Download Agent MUST NOT request user confirmation.

If the Download Descriptor includes the `suppressUserConfirmation` element and the value is equal to 'Always' and the Download Server can not be authorized then the Download Agent MUST request user confirmation prior to proceeding.

If the Download Descriptor includes the `suppressUserConfirmation` element and the value is equal to 'UserConfirmStepOnly' and the Download Server can not be authorized then the Download Agent MUST request user confirmation prior to proceeding.

If the `suppressUserConfirmation element` is only defined at a Product level then the Download Agent MUST apply the value of the `suppressUserConfirmation` element to all the Media Objects that belong to the Product.

If the `suppressUserConfirmation` element is defined at the Media Object level (i.e. within individual `mediaObject` elements) then the Download Agent MUST only apply the value of the `suppressUserConfirmation` to the Media Object that it is specified for.

If the `suppressUserConfirmation` element is defined for a Product and also for individual Media Objects that belong to that Product then the Download Agent MUST as a default apply the value of the `suppressUserConfirmation` element defined for the Product to all the Media Objects specified within that Product, however if the `suppressUserConfirmation` element is also defined for an individual Media Object within that Product then this value takes precedence and overrides the value defined for the Product.

If the Download Descriptor includes the `suppressUserConfirmation` element and the value is equal to 'Never' the Download Agent MUST request user confirmation prior to proceeding.

If the Download Descriptor does not include the `suppressUserConfirmation` element the Download Agent MUST treat the Download Descriptor as if it includes the `suppressUserConfirmation` element with the value equal to 'Never'.

If, based on the value of the `suppressUserConfirmation` element(s) the Download Agent determines that it needs to request user confirmation then the following information SHOULD, if available, be presented to the user:

- `name`
- `vendor`
- `size`
- `type`
- `description`
- `downloadTime`
- Additional defined textual metadata

If the Download Descriptor determines that the device does not support one or more of the media types as specified by the `type` element(s) then the behaviour of the Download Agent is dependant upon whether user confirmation is required or not:

- If user confirmation is required (i.e. the server can not be authorized (see section 8.3) or the Download Descriptor does not contain the `suppressUserConfirmation` element or it is present and the value equal to 'Never') then the user SHOULD be prompted to confirm if they wish to proceed with the Download. If the user chooses to reject or cancel the download, based on the information presented, the Download Agent MUST post a "User Cancelled" status report. In case of a Compound Product, the Download Agent MUST NOT show to the user the separate Media Objects of the Compound Product, but MUST show the Compound Product as a single entity to the User.

- If user confirmation is not required (i.e. the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to ''Always' or 'UserConfirmStepOnly' and the Download Server can be authorized, see section 8.3) then the Download Agent MUST post an "Invalid Type" status report.

The `type` element may occur multiple times for each Media Object. This indicates that the device is recommended to support all the listed media types in order to successfully download and use the complete Media Object. In this case the Download Agent should interpret the multiple `type` elements in a manner such that the order of occurrence indicates the decreasing importance to the user, i.e. that first `type` element has the highest relevance for presentation (it would typically be the, one or more, most important media types to be rendered or executed). If the Media Object is packaged or wrapped in a container format then the first, one or more, `type` elements would represent the innermost Media Objects.

5.2.4 Step 5a; Media Object retrieval

The retrieval of Media Objects is typically performed using HTTP [RFC2616] but always according to the scheme in the `objectURI` elements of the Download Descriptor. The Download Agent MUST at a minimum support HTTP [RFC2616] and TLS [RFC2246] and MAY support WSP [WSP] or WTLS [WTLS], as well as other optional protocols. For more details on specifics on the use of HTTP see section 5.5 HTTP Specific Functionality.

The request for a Media Object MUST be for exactly the URI specified in the descriptor, but the request MAY include additional headers created by the Download Agent.

The mechanics of this action is explained more in detail in specifications covering [RFC2616] and [WSP].

If the Download Descriptor contains multiple Media Objects, the Download Agent MAY retrieve each Media Object sequentially or in parallel. The exact algorithm for determining the order of downloads, or for deciding to perform downloads in parallel is outside the scope of this specification.

If a Media Object does not exist then the Download Agent MUST post a “Loader Error” status report.

If the user aborts the retrieval of a Media Object then the Download Agent MUST post a “User Cancelled” status report.

If the Download Descriptor contains a Compound Product, and if any of the above errors situations occur for one or more of the Media Objects of the Compound Product, the Download Agent MUST discard all the Media Objects downloaded for the Compound Product, except when the Download Agent and Download Server support pause and resume, and the error condition is a temporary error (“Insufficient memory” or “Loss of Service”). In this latter case, the Download Agent MAY continue as defined in section 5.2.4.3 Pause and Resume Media Object Retrieval.

5.2.4.1 Media Object Retrieval of Download Timing reservation

OMA DLOTA version 2.0 supports the function of timing reservation of the Media Objects retrieval. The Download Agent SHOULD support this function. The Download Server SHOULD support this function. The Download Agent MUST advertise support for this function using UAProf header. If the `downloadTime` element is present in the Download Descriptor, the Download Agent SHOULD execute the download transactions for the Products defined in the Download Descriptor at the time designated by the `downloadTime` element. If the Download Agent was able to set the timer at the time designated by the `downloadTime` element and if the `reservationNotifyURI` element is present in the Download Descriptor, then the Download Agent MUST post a “Set Timer Success” status report to the `reservationNotifyURI`. The `size` element and `installSize` element MAY be equal to zero if the `size` or `installSize` is not known when the timing reservation Download Descriptor is Downloaded. If the `size` element or `installSize` element is equal to zero, the `updateDDURI` element MUST be included and the Download Agent MUST fetch the new Download Descriptor from the `updateDDURI` to obtain the actual size of the Media Objects or Products. If the `timeInterval` element is present in addition to `downloadTime` element in the Download Descriptor, then the Download Agent SHOULD execute the download transactions for the Products defined in the Download Descriptor with the desired time interval originated at the time specified in the `downloadTime` element till the time specified in the `timeIntervalExpire` element. The Download Agent SHALL NOT execute the download transactions at times later than that specified in the `timeIntervalExpire` element.

The process for negotiating the download time between the User and the Download Server is outside the scope of this specification. For example, the User can use a web page to select the desired download time prior to initiating the download process. The Download Agent SHOULD execute the download transaction at the designated time if the `timestamp` element is present in the Download Descriptor.

Before the Download Agent executes the download transaction at the designated time, if the Download Descriptor contains `updatedDDURI` element, the Download Agent SHALL initiate the updating Media Object retrieval as defined in section 5.2.4.2 Media Object Retrieval of Updated Media Object. If the Download Agent determines that the Media Object has been replaced or updated, the Download Agent MUST proceed as defined in section 5.2.4.2.

If the Download Agent determines that the `downloadTime` element in the Download Descriptor, available at the `updateDDURI`, has been updated to a time in the future, the Download SHALL NOT download any Products or Media Objects. The Download Agent SHOULD execute the download transaction defined by the new `downloadTime`. If the Download Agent was able to set the timer at the time designated by the updated `downloadTime` element and if the `reservationNotifyURI` element is present in the updated Download Descriptor, then the Download Agent MUST post a “Set Timer Success” status report to the `reservationNotifyURI`. If `timeInterval` and/or `timeIntervalExpire` elements are updated, the new values MUST be used for all succeeding download transactions.

If the user aborts the automatic download after the Download Agent is ready to set the download transactions (e.g. after the Download Agent set a internal timer to execute the download transaction), and if the `reservationNotifyURI` element is present in the Download Descriptor, then the Download Agent MUST post a “Reservation Cancelled ” status report to the `reservationNotifyURI`.

If the `timeInterval` element is present and a download transaction is not concluded before the next download transaction (from the same Download Descriptor) shall be is initiated from the same Download Descriptor, then the Download Agent MUST discarded the new download transaction to avoid that the updating of Products or Media Objects gets updated before the Products or Media Objects before they have been downloaded.

If the Download Agent is not able to execute the download transactions at the designated time, and if the `reservationNotifyURI` element is present in the Download Descriptor, then the Download Agent SHOULD post a "Reservation Error" status report to the `reservationNotifyURI`. If the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT present a message to the User, in all other cases the Download Agent SHOULD present a message to the User that the reserved downloading was not executed.

If the Download Agent is not able to execute the download transaction at the designated time, due to the fact the device was switched off, and the if the `reservationNotifyURI` element is present in the Download Descriptor, then the Download Agent MAY post a "Reservation Error" status report to the `reservationNotifyURI` or make a new download attempt. If the `timeInterval` element is present and it has not been possible to execute one or more several download transaction(s) have not been possible to execute at the designated time, then the Download Agent MUST NOT send more that one "Reservation Error" status report per Download Descriptor or the Download Agent MUST only initiate a new download attempt for the latest time interval.

If the Media Object is no longer available then the Download Server MUST return an HTTP 410 "Gone" status code in response to either a request for the actual Media Object or a request to the `updatedDDURI`. Upon reception of an HTTP response with this status code the Download Agent MUST permenantly remove the reservation and send status message 973 reservation cancelled if the `reservationNotifyURI` element is present in the Download Descriptor. If the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT present a message to the User, in all other cases the Download Agent SHOULD present a message to the User that the reserved download is no longer available.

The Download Agent SHOULD NOT send the status codes other than the Set Timer Success (971), Reservation Error (972) and Reservation Cancelled (973) status report to `reservationNotifyURI`. The Download Agent SHOULD use the `installNotifyURI` to send such error status codes.

See section 6 Status Report Functionality for details on sending reservation related notifications.

5.2.4.2 Media Object Retrieval of Updated Media Object

DLOTAv2 includes the functionality to replace or update Media Object(s) which are available and installed on the device. There are two ways to update the Media Object. The first mechanism makes use of the `objectID` and `objectVersion` elements. The second mechanism uses HTTP ETag.

- Media Objects update by using `objectID` and `objectVersion` elements

This mechanism uses `objectID` and `objectVersion` elements to enable client based updating verification. The Download Agent SHOULD support this mechanism. The Download Server SHOULD support this mechanism.

When a Media Objects update is started, the Download Agent MUST retrieve the Download Descriptor of the Media Object(s) to be used as specified by the `updatedDDURI` element. The Download Server SHALL return the Download Descriptor. The Download Agent MUST discard the retrieved object if it is not a Download Descriptor. If `updatedDDURI` is not present in the Download Descriptor, the Download Agent SHALL NOT attempt to update the Media Object(s). If the Media Object is no longer available then the Download Server MUST return an HTTP 410 "Gone" status code in response to a request to the updatedDDURI. Upon reception of an HTTP response with this status code the Download Agent MUST cancel the update and MUST NOT make any further attempts to update the Media Object. If an `installNotifyURI` element is present in the Download Descriptor then the Download Agent MUST post an "Loader Error" status report. If the Download Descriptor contains the suppressUserConfirmation element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT present a message to the User, in all other cases the Download Agent SHOULD present a message to the User that the Media Object is no longer available and can not be updated.

This request SHOULD be triggered by the user or it may be triggered as a result of a download timing reservation. This request MAY be sent without user confirmation if the server can be authorized (see section 8.2) and the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to "Always". This request MAY also be sent without user confirmation if it is triggered for the download timing reservation (see section 5.2.4.1).

When performing an update the Download Agent MUST compare the value of the `objectID` element of existing Media Object and the value of the `objectID` element in the newly retrieved Download Descriptor. If the both values of the `objectID` elements are identical, then the Download Agent MUST compare the value of the `objectVersion` elements of the both. If the Download Descriptor contains the `suppressUserConfirmation` element with the value set to 'Always' and the server can be authorized (see section 8.3):

- The user MUST NOT be notified and:
    - If the Media Object version is greater than that of the existing Media Object the Download Agent MUST perform the necessary tasks to update the Media Object.
    - If the Media Object version is the same or less than that of the existing Media Object then the Download Agent MUST NOT download the Media Object and MUST post a 958 "Version Already Available" status report to the `installNotifyURI.`

In all other cases:

- The Download Agent MUST notify the user if the Media Object is a newer, older or same version of the existing `objectID` of the Media Object and MUST get confirmation from the user before proceeding.
- If the Download Agent gets confirmation from user to replace or update the Media Object, the Download Agent MUST retrieve the Media Object. Before retrieving the Media Object, the Download Agent MUST verify the available capabilities as defined in section 5.2.2.

To enable updating of a Media Object, the Server MUST include both `objectID` and `objectVersion` elements in `mediaObject` element of the Download Descriptor. The Server can choose whether to enable this functionality for each Media Object.

If the Download Descriptor contains multiple `mediaObject` elements or multiple `products` elements and the Download Agent supports the capability to download multiple Media Objects grouped within a single Download Descriptor, the Download Agent MUST compare its `objectID` and `objectVersion` one by one for each Media Object and SHOULD retrieve and install only updated Media Objects. This may help to reduce unnecessary download transaction to be executed.

If the Download Descriptor contains the `suppressUserConfirmation` element with the value set to 'Always' and the server can be authorized (see section 8.3) and the Download Descriptor contains multiple Media Objects that can be updated the user MUST NOT be notified, the Download Agent MUST take each Media Object in turn and:

- If the Media Object version is greater than that of the existing Media Object the Download Agent MUST perform the necessary tasks to update the Media Object.
- If the Media Object version is the same or less than that of the existing Media Object the Download Agent MUST NOT perform the necessary tasks to update the Media Object.

In all other cases, the Download Agent MAY offer the user the chance to select which Media Objects or Products that should be updated.

If the Media Object with higher `objectVersion` is successfully installed as defined in section 5.3, then the Download Agent MUST remove the old Media Object with lower `objectVersion`. The old Media Object with lower `objectVersion`, MUST NOT be removed until the new Media Object with higher `objectVersion` number has been successfully installed as defined in section 5.3.

- Media Object updating by using HTTP ETag

This OPTIONAL mechanism uses HTTP ETag header as defined by [RFC2616] to enable server based updating verification. The Download Agent MAY support this mechanism. The Download Server SHOULD support this mechanism.

The Download Server MUST expose the capability of supporting this mechanism by including the ETag header in the HTTP reply of the Download Descriptor if the Download Server supports this mechanism. The Download Server MUST include `updatedDDURI` in the Download Descriptor if it permits to use the Media Objects updating mechanism.

The Download Agent MUST persistently store the ETag value together with the `updatedDDURI` if the Download Agent supports this mechanism.

The Download Agent MAY send the HTTP If-None-Match header with the persistently stored ETag value from the already retrieved Download Descriptor (i.e. the old Download Descriptor) if the Download Agent want to confirm whether it is the newest version or not. The Download Agent MUST use the `updatedDDURI` to send the HTTP request.

This request SHOULD be triggered by the user, but this request MAY also be triggered automatically for example the download timing reservation function is used. This request MAY be sent without user confirmation if the server is an authorized server (see section 8.2) and the `suppressUserConfirmation` element is present in the DD and the value is "Always". This request MAY also be sent without user confirmation if it is triggered for the download timing reservation as specified in section 5.2.4.1.

The Download Server MUST compare the ETag value with the newest version of the Download Descriptor if the Download Server receives the HTTP If-None-Match header and the Download Server supports this mechanism. The Download Server MUST send the new Download Descriptor to the Download Agent if the comparison result shows the old Download Descriptor is updated. The Download Server MUST send the HTTP status code (304 Not Modified) to the Download Agent if the comparison result shows the old Download Descriptor is not updated. If the Media Object is no longer available then the Download Server MUST return an HTTP 410 “Gone” status code in response to a request to the `updatedDDURI`. Upon reception of an HTTP response with this status code the Download Agent MUST cancel the update and MUST NOT make any further attempts to update the Media Object. If an `installNotifyURI` element is present in the Download Descriptor then the Download Agent MUST post an “Loader Error” status report. If the Download Descriptor contains the `suppressUserConfirmation` element and the value is equal to ‘Always’ and the server can be authorized (see section 8.3) the Download Agent MUST NOT present a message to the User, in all other cases the Download Agent SHOULD present a message to the User that the Media Object is no longer available and can not be updated.

If the Download descriptor is updated, the Download Agent must apply the procedures defined for Media Object updating by using `objectID` and `objectVersion` elements received in the updated Download Descriptor.

If the Media Object with higher `objectVersion` is successfully installed as defined in section 5.3, then the Download Agent MUST remove the old Media Object with lower `objectVersion`. The old Media Object with lower `objectVersion`, MUST NOT be removed until the new Media Object with higher `objectVersion` number has been successfully installed as defined in section 5.3.

5.2.4.3 Pause and Resume Media Object Retrieval

A Download Server SHOULD support the capability of pause and resume retrieval of Media Objects using the HTTP Range Retrieval mechanism as defined in [RFC2616]. If the Download Server supports the capability to pause and resume retrieval of Media Objects, this capability MUST be exposed by the Download Server in the Media Object response using the HTTP header *Accept-Ranges* with the *range-unit* value set to *bytes* as defined in [RFC2616]. In addition, if the Media Object has an `objectId` and an `objectVersion` element in the associated Download Descriptor, the ETag header in the Media Object response MUST have the following value:

“ETag” “:” objectID”/”objectVersion

objectID=< Syntax as defined in section 7.2.3.3.5 by the objectId element in the Download Descriptor>

objectVersion=< Syntax as defined in section 7.2.3.3.6 by the objectVersion element in the Download Descriptor>

A Download Agent SHOULD support pause and resume object retrieval. Media Object retrieval can be paused or interrupted for several reasons. The end user might decide to postpone the download, the network connectivity is lost, there is no memory available or the device gets out of power. It is out of scope for this specification to define which events that can pause or interrupt an on going Media Object retrieval.

If the Download Agent terminates the transport connection or detects that the transport connection is lost for some reason, the Download Agent SHOULD NOT discard the received content range if the server has exposed the support for Range Retrieval with the *Accept-Ranges* header. The received content range shall be stored in device but it SHALL NOT be available until before the remaining parts of the content have been downloaded and the content has been successfully installed.

If the end user or Download Agent decides to cancel a paused Media Object Retrieval, the Download Agent MUST post a “User Cancelled” status report, if the `installNotifyURI` element is present.

Resumption of Media Object Retrieval can be triggered by several events. The user might resume the download manually or the network connectivity might be restored. It is out of scope for this specification to define the policy for the events that should resume a paused Media Object retrieval.

Media Object Retrieval is resumed by the Downloading Agent by sending a new GET request to the `objectURI`. The Range header MUST specify the remaining content range that has not been downloaded. The Download Agent MUST NOT retrieve more than one content range. The If-Match header MUST contain the value of the Etag header provided in the paused Media Object retrieval if the ETag value is available.

If the Media Object is no longer available then the Download Server MAY return either an HTTP 410 "Gone" or an HTTP 404 "File Not Found" status code. Upon reception of an HTTP response with either of these status codes the Download Agent MUST cancel the attempt to resume the download of the Media Object and MUST NOT make any further attempts to resume the download of the Media Object. If an `installNotifyURI` element is present in the Download Descriptor then the Download Agent MUST post an "Loader Error" status report. If the Download Descriptor contains the suppressUserConfirmation element and the value is equal to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT present a message to the User, in all other cases the Download Agent SHOULD present a message to the User that it was not possible to resume the download of the Media Object.

If the Download Agent receives an HTTP Status Code, 412 Precondition failed. The Media Object has been updated since the Media Object retrieval was paused. The Download Agent MUST post a "Media Object Updated" status report if the `installNotifyURI` element is present and discard the saved content range. If the Download Descriptor contains the `suppressUserConfirmation` element with the value set to 'Always' and the server can be authorized (see section 8.3) the Download Agent MUST NOT notify the user. In all other cases the Download Agent MUST notify the end user. The Download Agent MUST then start a Media Object Update retrieval procedure, as defined in section 5.2.4.2 by downloading the new Download Descriptor from the `updatedDDURI`. If `updatedDDURI` is not present in the Download Descriptor, the Download Agent SHALL NOT perform a Media Update Object retrieval procedure. Once the new Download Descriptor is downloaded to the device, the Download Agent SHALL continues with "Capabilities Check" as defined in section 5.2.2.

If the Download Agent receives an HTTP Status Code 206 Partial Content, the client SHALL concatenate the saved content range with the received content range and continue with the installation process as defined in section 5.3.

A Download Agent MUST NOT attempt to resume the download of Media Object unless the Download Server has exposed this capability with the inclusion of the *Accept-Ranges* header as specified above.

If the Download Descriptor contains multiple `mediaObject` elements, multiple `product` elements or multiple `license` elements that have not been downloaded, the Download Agent MUST continue to download the remaining Products, Media Objects or Licenses once the Downloading session is resumed. The Download Agent MUST NOT allow more than 5 automatic resume retries. If the threshold is exceeded, the Download session SHALL be interrupted and the Downloaded Media Objects and Licenses SHALL be discarded. The user MAY retry to resume the Download transaction without any limitation.

5.2.4.4 Object Retrieval of chunked Media Objects

A Download Agent and a Download Server MAY support the capability of retrieval of chunked Media Objects using the HTTP Range Retrieval mechanism as defined in [RFC2616]. The HTTP Range Retrieval mechanism can be used in order to enable the download of Media Objects that are larger than the maximum transfer size that may be imposed by the underlying network.

The Download Agent MUST NOT send the successful Download Completion Notification (i.e. Download Completion (981)) unless the Download Agent can complete the download of the entire Media Object. The Download Agent MUST NOT send the successful Install Notification (i.e. Success (900)) unless the Download Agent can successfully install the entire Media Object. The Download Agent MAY send an appropriate error status code (as defined in section 6) if the Download Agent encounters an error or the download is interrupted.

In the case where a Download Server does not support Range Retrieval but a Download Agent has requested Range Retrieval the Download Server shall return the complete Media Object. In this case, the Download Agent MUST immediately abort the download of the Media Object if the Download Agent does not have enough resources to retrieve the entire Media Object and the Download Agent MUST send the Insufficient Memory (901) status report if the `installNotifyURI` element is present in the associated Download Descriptor.

5.2.4.5 Media Objects Retrieval from Multiple Servers

If a Download Descriptor includes multiple `server` elements in an `objectURI` element, the Download Agent MAY fetch the Media Object(s) or parts thereof (using the the HTTP Range Retrieval mechanism as defined in [RFC2616]) from multiple DLOTA Servers simultaneously. The Download Agent reconstructs Media Object(s) as soon as the Download Agent receives data from a subset of the DLOTA Servers.

5.2.5 Step 5b; License Retrieval

If a Media Object or Product has a `license` element, the Download Agent MUST pass the contents of the `license` element transparently to the appropriate License Agent. The License Agent MUST retrieve the appropriate License and report back to the Download Agent the result of the License retrieval. The Download Agent MUST include the License retrieval status in any report posted to the server. It should be noted that it is possible to use a Download Descriptor for the purpose of license upgrade i.e. when the device has already downloaded the Media Object. In this case the Download Descriptor will not contain any `mediaObject` elements (see example D.9).

If the device supports OMA DRMv2 [OMADRMv2], it MUST support the license element and associate functionality.

If the License Retrieval status fails with a permanent error, the Downloading of the Media Object or Product associated with the `license` element MUST be interrupted and the Media Object and License information MUST be removed from the device.

If the License Retrieval status fails with a temporary error and if pause and resume is supported by the Download Agent as well as the Download Server, the Download transaction SHALL be paused and the end user SHALL be informed about the reason for pausing the Download transaction. The Download Agent MAY offer the end user to resume the Download transaction immediately, keep the Downloaded Media Object, Media Content Range or Product for later resumption of the Download transaction or define a time when the Download transaction shall be resumed. If `suppressUserConfirmation` is equal to true and the Download Server is authorized as defined in section 8.3, the Download session SHALL NOT be automatically paused during a temporary error. The Download Agent SHALL continue to download as much as possible. If pause and resume is not supported, a temporary error SHALL be treated with same procedures as for a permanent error.

If the Download Descriptor contains a Compound Product and if a permanent error occur or if the Download Agent do not support pause and resume, the Download Agent MUST discard all the Media Objects downloaded and Licenses for the Compound Product.

If a download reservation (Section 5.2.4.1) is defined for the Download Descriptor, this reservation MUST also be used for determining when to retrieve the License.

The Download Agent MAY perform Media Object download and License retrieval in sequence or in parallel. In case of sequential retrieval, the License referenced by first `license` element MUST be retrieved before for the downloading of the Media Object. If the License is associated with a Product and sequential downloading is applied, the License referenced by the first `license` element MUST be retrieved before the first Media Object of the Product is downloaded. In case of parallel Media Object download and License retrieval, the retrieval of the License of the first `license` element MUST start before or at the same time as the Downloading of the Media Object. If the License is associated with a Product and parallel downloading is applied, the License referenced by the first `license` element MUST be retrieved before the first Media Object of the Product is downloaded.

If the `license` element includes the `order` attribute and the `order` attribute is equal to the `post`. The contents of the `license` element MUST NOT be passed to the License Agent unless the Media Object or Product is successfully downloaded as defined in section 5.3.2. If the `order` attribute is equal to `post`, the Download Agent MUST continue with the next `license` element. The exact algorithm for Downloading of the remaining Licenses is not defined in this specification.

If the Download Agent supports Progressive Download and receives a Download Descriptor which contains one or more `license` elements associated with one or more Media Object elements that contains a `progressiveDownloadFlag` element with the value “true”, the Download Agent MUST NOT make the Media Object(s) available for rendering before the associated License(s) have been successfully installed. The Download Agent MUST include the License retrieval status in

any report posted to the server. The exact algorithm for determining the number of Media Objects and the order in which they shall be progressively downloaded is outside the scope for this specification.

5.2.5.1 Maintaining Media Object and License Consistency

If the device contains a License Agent and attempts to update or delete a Media Object (that was retrieved as a result of processing a Download Descriptor that contained a License element) then:

- If the Media Object was updated successfully, as defined in section 5.2.4.2 , the Download Agent MUST inform the appropriate License Agent (as identified by the `lictype` attribute) that the Media Object has been updated.
- If the user wants to update a Media Object that was retrieved as a result of processing a Download Descriptor in which the Media Object was part of a Product or Compound Product then the Download Agent MUST attempt to update all the Media Object(s) in the Product or Compond Product using the process described in 5.2.4.2.
- If the Media Object was deleted from the device, the Download Agent SHOULD inform the appropriate License Agent (as identified by the `lictype` attribute) that the Media Object has been removed.

5.2.6 Step 6; Sending Download Completion Notification

The purpose of the Download Completion Notification is to provide the Status Report Server with an indication that a Media Object or Product was downloaded successfully. At this point in the download process the Media Object or Product is not available to the user. The Download Completion Notification is OPTIONAL. Therefore, the Download Server may not receive the Download Completion Notification even if the Media Object or Product is successfully downloaded.

This step is valid only in the cases where it has been explicitly requested. This is indicated by including the `downloadNotifyURI` element in the Download Descriptor.

If the `downloadNotifyURI` element is present in the Download Descriptor and the Download Agent support the Download Completion Notification function, the Download Agent MUST send the Download Completion Notification (981) status report to the address defined in the `downloadNotifyURI` element. If the `downloadNotifyURI` is defined for a Product, the Download Agent MAY send the Download Completion Notification (981) status report after the completion of retrieving all the Media Objects specified for that Product in the Download Descriptor. If the `downloadNotifyURI` element is missing from the Download Descriptor then no Status Report can be sent.

If Download Completion Notification is defined for a Product, Download Completion Notification MUST NOT be specified for separate Media Objects. The Download Agent MUST discard any notification request for separate Media Objects if Download Completion Notification is defined for a Product.

If a Product is a Compound Product, the Download Completion Notification MUST NOT be specified for separate Media Objects of the Compound Product. It MAY only be specified for the Product as a whole.

The Download Agent SHOULD NOT send the status codes other than the Download Completion Notification (981) and Mixed Status (970) status report to `downloadNotifyURI`. The Download Agent SHOULD use the `installNotifyURI` to send such error status codes.

The process for sending the Download Completion Notification is the same as that for sending the install notification as explained in section 5.3.2.

5.3 Object Installation Process

Object installation is the process by which Media Objects made available for execution or rendering.

To install Media Objects, the Download Agent MUST also use the Status Report mechanism (if requested in the Download Descriptor) to give the server infrastructure feedback about a possible failure of the installation event, and MUST use the mechanism to report on a successful installation. For details related to the Status Report mechanism see section 6 Status Report Functionality.

5.3.1 Step 7; Installation

The installation is a media type specific, and implementation specific, mechanism to prepare Media Objects for rendering/execution on the device.

The functionality of this step "Installation" depends on whether the Installation Notification has been requested. This is indicated by including the `installationNotifyURI` element in the Download Descriptor.

The Download Agent MUST finish both step 5a Object Retrieval and step 5b License Retrieval for Media Objects or Products that have a `license` element before starting the installation step.

In case of `installationNotifyURI` element is defined for a Compound Product the Download Agent MUST NOT start the installation of the Media Objects in the Compound Product, before all Media Objects and Licenses of the Compound Product are ready to be installed.

Installation Notification NOT requested

In case an Installation Notification has not been requested the download use case is terminated when the installation has completed. The Media Objects can now be rendered or executed.

Installation Notification requested

In case an Installation Notification has been requested by including the `installNotifyURI` element in the Download Descriptor, then the installation process is split into two phases.

- The first phase (covered as step 8 in this specification) consists of a pre-installation where the device prepares the Media Objects for rendering/execution to the largest extent possible without actually allowing it to be used. In case of a Product, this phase MUST be done for all the Media Objects of the Product before starting the second phase.

- The second phase is dependant on the success of the next step (covered as step 9 in this specification), the execution of the Installation Notification. The Media Object(s) will only be made available for execution/rendering if that step is regarded as successful, however, if the Download Descriptor contains the `progressiveDownloadFlag` element with the value set to 'true' then the Media Objects can be made available for rendering as the Media Objects are being downloaded.

These two phases are valid also in case the Download Descriptor and the Media Object(s) are co-delivered to the device.

Installation is complete when the Media Object or Product has been prepared for execution/rendering on the device, or the Download Agent encountered an error. In either case, the status MUST be reported (assuming the installation notification has been requested as indicated by the `installNotifyURI`) as described in section 6 Status Report Functionality.

- If the installation succeeded the device MUST generate the status code "Success" in the Status Report as described in Section 5.3.2 "Step 8, Sending Installation Notification".

- If the user cancelled the downloading and installation before the installation completed (i.e. before the sending of the installation notification was completed) the device MUST generate the status code "User Cancelled" in the Status Report.

- If the installation of the Media Object or Product failed due to lack of memory then the device MUST generate the status code "Insufficient Memory" in the Status Report.

- If the installation of the Media Object or Product failed due to a reason independent of the end user, and some other reason, than lack of memory then the device MUST generate the status code "Device Aborted".

- If the Downloading User Agent immediately rejected a Media Object because it had characteristics that made it impossible to execute or render on the device then the device MUST generate the status code "Non-Acceptable Content" in the Status Report.

- If the Downloading User Agent could not install a Media Object because the elements of the retrieved Media Object differed from the elements defined in the Download Descriptor then the device MUST generate the status code "Attribute Mismatch".

In all cases where a Status Report indicating an error is reported the Media Object, or in case of a Compound Product all Media Objects of the Compound Product, MUST be discarded by the device except when the Download Agent and Download Server supports pause and resume Media Object retrieval as define section 5.2.4.3 and it is an temporary error ("Insufficient memory" or "Loss of Service").

Figure 2 shows the installation process. The successful completion of the download transaction is independent of the content handler that will finalize the processing of the retrieved media type. However, the Download Agent and the approriate content handler may negotiate during the transaction about the device capabilities to process the content.

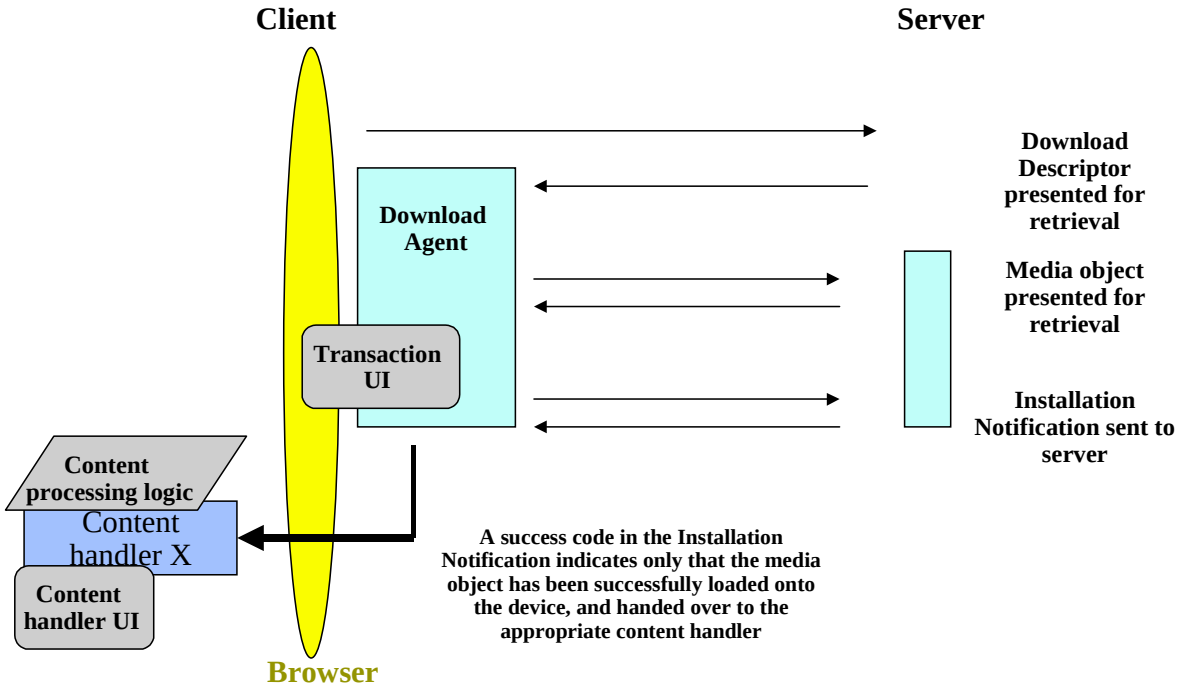

Figure 2: Installation Process

5.3.1.1 Media Object installation parameter

The element `installParam` MAY be used to convey installation parameters to the Media Type specific Media Object Installer. If an `installParam` exists then the Download Agent MUST pass the value of the `installParam` to the Media Object Installer.

If the Media Object installer does not support the installation parameter, the installation parameter must be ignored. If the Media Object installer supports the installation parameter, but there is an error in the parameter, then an error code ("attribute mismatch") SHOULD be returned as defined in section 5.2.4.

If the Media Object is a Composite Object then the `installParam` parameter is conveyed by the Download Agent to the Media Object Installer of the Composite Object. The use of the value of this parameter is specific to the Media Object Installer and outside the scope of this specification.

5.3.1.2 Progressive Download

It is RECOMMENDED that devices support progressive download. If the device support progressive download and a Download Descriptor contains the `progressiveDownloadFlag` element with the value set to 'true' then where possible the Download Agent SHOULD make the Media Objects available for rendering as the Media Object is being downloaded.

As specified in section 5.5.1.2 if the device supports progressive download then it MUST advertise the mime/media types that can be progressively downloaded using the OMADownloadProgressiveDLAccept header. If the Download Descriptor contains an `installNotifyURI` element for a Media Object that also has `progressiveDownloadFlag` element with the value set to 'true' then the Download Agent MUST attempt to send an installation noftification once the Media Object(s) have been downloaded, if this attempt fails the Download Agent MUST delete the Media Object(s).

5.3.2 Step 8; Downloading of Post Licenses

If the Media Objects or Products has been successfully installed as defined in section 5.3.1, and if the Media Objects or the Products have one or many `license` elements with the `order` attribute equal to `post`, the Download Agent MUST pass the contents of those `license` elements transparently to the appropriate License Agent. The License Agent must retrieve the appropriate License and report back to the Download Agent the result of the License retrieval if `installNotifyURI` element is defined for the Media Object. If `installNotifyURI` element is not defined for the Media Object or Product, no status will be reported. If the License Retrieval status fails with a permanent error, the Media Objects and License information associated with the `license` element MUST be removed from the device.

If the Download Descriptor contains a Compound Product, and if any of the above errors situations occur for one or more of the License of the Compound Product, the Download Agent MUST discard all the Media Objects downloaded and Licenses for the Compound Product.

If the Media Objects or Products has not been successfully installed as defined in section 5.3, the Download Agent MUST not pass the contents of the `License` elements to the appropriate License Agent.

5.3.3 Step 9; Sending Installation Notification

The purpose of the Status Report mechanism is to provide the download server with an indication that the Media Object or Product has been properly received and installed. This functionality is available as the success or failure of the installation of a Media Object or Product may be critical to execute certain business models within the content service realm.

This step is valid only in the cases where it has been explicitly requested, that is when the `installNotifyURI` element is included in the Download Descriptor for a Product or a single Media Object.

If the `installNotifyURI` element is present in the Download Descriptor for one or many Media Objects, the Installation Notification status report MUST be sent to the address defined in the `installNotifyURI` element for each Media Object. If the `installNotifyURI` element is present for a Compound Product, the Installation Notification status report MUST be sent after the completion of retrieving all the Media Objects specified in the Download Descriptor for the Compound Product. If the `installNotifyURI` element is missing from the Download Descriptor for the Media Object and for the Product containing the Media Object, then no Status Report can be sent.

The installation status is reported by the use of the schema defined in the `installNotifyURI` element. If the scheme is HTTP or HTTPS then an HTTP POST request to the defined URL MUST be performed. However, if the transfer of the Media Object has been performed using a different scheme than HTTP then the Download Agent MUST also be able to execute the Installation Notification using this same scheme.

In case the defined scheme of the Installation Notification is HTTP or HTTPS then:

- The content of the body of the POST request MUST include on the first line a status code and status message. The table "Installation Status Code and Associated Message" in Section 6.1 DLOTAv2 Status Reports lists the valid status message codes and messages. The Section 6.2 Status Report Formatting defines the format of the installation notification.

- In case the `installNotifyURI` is specified for a Media Object or Product that has a `license` element and the license installation has not been already reported as defined in section 5.2.4.5 during progressive download, a second line MUST be added containing the License status code and status message. The status code reported MUST be either 956 (License Retrieval Succeeded) or 957 (License Retrieval Failed). The Section 6.2 defines the format of the installation notification.

- In case the `installNotifyURI` is specified for a Product, the Download Agent MAY add additional status lines for each Media Object defined for the Product. The Download Agent MUST add additional status lines for Media Objects defined for the Product if the status of all the Media Objects for the Product are not the same. In such case, the overall installation status SHOULD be set to 970 "Mixed Status" error. Also the Download Agent MUST add an additional License retrieval status lines for each Media Object of the Product that has a `license` element if the license installation has not been already reported as defined in section 5.2.4.5 during progressive download.

- The sending of the Installation Notification is regarded as successful if the Download Agent receives an HTTP-Response from the Status Report Server with any 200-series response code. All other HTTP-response codes (100-, 300-, 400- and 500-series HTTP-response codes) MUST be handled as either temporary or permanent errors. The Download Agent MUST implement the behaviour associated with response codes representing temporary errors (for example "401" and "407") as defined in [RFC2616].

- No content body should be returned in the HTTP response, if any is sent, the Download Agent MUST ignore the body part.

- If a request brings no response, the request MAY be retried, but it SHOULD NOT be retried if any response is received (except in case, e.g. "401 unauthorized", the reply prompts a modified retry).

- The exception to the logic defined above is the semantics of a "Well-Intentioned Attempt". If a well-intentioned Installation Notification attempt brings no response from the Status Report Server then the Download Agent MUST equate the situation to the reception of a 200-series response code. This may for example occur in the situation when the Download Agent experiences a timeout before the response is received. The time to wait for the HTTP-Response is implementation specific.

For other schema:

- The sending of the Installation Notification is regarded as successful if a response (the format of which will be dependant on the schema) indicating successful reception of the Installation Notification is received from the Status Report Server.

- If a request brings no response, the request MAY be retried.

- If a well-intentioned Installation Notification attempt brings no response from the server then the Download Agent MUST equate the situation to the reception of a response indicating that a successful reception of the Installation Notification was received from the Status Report Server.

If the Download Descriptor contains the `installNofifyURI` element for a Media Object, then the Media Object MUST NOT be released for use at the device unless the sending of the installation notification succeeds. The exception to this rule is when the Download Descriptor contains a `progressiveDownloadFlag` element with the value set to 'true'. In this case, where possible, the Media Object should be made available for rendering as the Media Object is being downloaded. Once the Media Object has been successfully downloaded then the Download Agent must send the appropriate installation notification, if this fails the Media Object MUST be deleted. If the `installNotifyURI` is defined for a Product, then all the Media Objects of the Product MUST NOT be released for use at the device unless the sending of the installation notification succeeds. For Products and Compound Products if one or more of the Media Objects of product contain a `progressiveDownloadFlag` element with the value set to 'true' then, where possible, all Media Objects in the Product should be made available for rendering as the Media Objects are being downloaded. Once all the Media Objects have been successfully downloaded then the Download Agent MUST send the appropriate installation notification, if this fails all the Media Objects MUST be deleted.

If no well-intentioned attempt can be made then the device MUST NOT allow for the use of the Media Object(s).

If the Download Descriptor contains a `suppressUserConfirmation` element and its value is set to 'Always' and the server can be authorized then the device MUST NOT notify the user that the download failed and it MUST remove the Media Object from the device.

For all other cases the Download Agent MUST indicate to the user that the download failed and MUST remove the Media Object from the device.

5.3.3.1 Installation Notification Semantics

The Installation Notification mechanism should prevent a situation where the server has “knowledge” that the transaction completed, but the client perceives the transaction as failed (and so prevents the release a Media Object or Product for use in the device). This logic has to function properly also in situations where a proxy separates the Download Agent and the download server.

The Download Agent MUST make at least one well-intentioned attempt to send the Installation Notification to the address defined in the `installNotifyURI` element. By “well-intentioned” is meant an attempt where the network connection is known (to the extent possible) to be present. If such “well-intentioned” attempt cannot be made, despite multiple retries, then the Media Object or Product MUST immediately be removed from the device.

5.4 After the Object Installation Process

After the device sent the status report, or the download transaction completed without a Status Report (the Download Descriptor did not include an `installNotifyURI` element) then the Download Agent proceeds to the after process of the Media Object installation.

5.4.1 Step 10; Download Confirmation and Next Step

The functionality of this step “Download Confirmation and Next Step” is dependant on whether the Download Descriptor conatains a `suppressUserConfirmation` element (with the value set to ‘Always’) and if the server can be authorized (see section 8.3).

- If the Download Descriptor contains a suppressUserConfirmation element and its value is set to ‘Always’ and the server can be authorized:

  If the Download Descriptor contains a `nextURL` element then the Download Agent SHOULD open the browser at the nextURL.

- For all other cases:

  When the device has downloaded all post License elements, as described in step 8, sent the status report, as described in step 9 above, or the download transaction completed without a Status Report (the Download Descriptor did not include an `installNotifyURI` element) then the Download Agent SHOULD indicate the result of the transaction to the user.

  The download transaction may have ended successfully, or failed at any step of the download process. In both these cases the Download Agent SHOULD present the user with the option to either continue with a local context specific operation (often to render or execute the Media Object) or to continue with a browsing operation. The exception to this rule is when the server can be authorized (see section 8.2) and the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to "Always". In this case the Download Agent MUST NOT present the user with an option dialog, if the Download Descriptor also contains a `nextURL` element the Download Agent MUST invoke the URL defined in the `nextURL` element.

  If the end user selects to continue with a browsing operation then the Download Agent SHOULD invoke the URL defined in the `nextURL` element. The flow control functionality implemented using `nextURL` may link to any resource either in the download server (that may issue a context sensitive redirection), in the presentation server, or somewhere else.

  A context sensitive redirection provides an opportunity for a late binding for the control flow. This may be particularly important for error situations, to allow the presentation server to optimise its response based on the type of error that occurred.

  If the `nextURL` element is not present in the Download Descriptor then the Download Agent SHOULD offer the user an opportunity to continue with a local operation, and MAY offer the user an opportunity to continue with a browsing operation. The exception to this rule is when the server can be authorized (see section 8.2) and the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to "Always".

In this case the Download Agent MUST NOT present the user with an option dialog. In the case the end user selects to continue with a browsing operation the URL to be activated is implementation specific.

The Download Agent MAY allow the user to continue with a local context specific operation or to continue with a browsing operation by browsing to the `nextURL` before the Media Objects are downloaded and installed. In this case, the Download Agent can download the Media Objects in the background. Also in this case, the Download Agent SHOULD present the user an option to browse to the Support URI from the `vendor` element, if available, when downloading and installing the Media Objects fails in anyway, after sending the status report if the `installNotifyURI` is defined. The exception to this rule is when the server can be authorized (see section 8.2) and the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to "Always". In this case the Download Agent MUST NOT present the user with an option dialog.

5.4.2 Step 11; Sending Deletion Notification

The purpose of the Deletion Complete Notification is to provide the download server with an indication that the Media Object has been removed from the device. The sending of the Deletion Complete Notification is valid only in the cases where it has been explicitly requested, that is when `deleteNotifyURI` element is included in the Download Descriptor. The Download Agent MAY support this function. The Download Server MAY support this function.

If the `deleteNotifyURI` element is present in the Download Descriptor for a Media Object, then the Download Agent MAY send the Deletion Complete Complete (991) status report to the URI specified in the `deleteNotifyURI` element. The delete notification URI MUST only be used for single Media Objects, In case the delete notification URI is defined for a Product, the Download Agent MUST discard the request

The Download Agent SHOULD NOT send the status code other than the Deletion Complete (991) status report to the URI specified in the `deleteNotifyURI` element. The Download Agent SHOULD use the `installNotifyURI` to send such error status codes.

The process for sending the Deletion Complete Notification is the same as that for sending the Install Notification as explained in section 5.3.2.

The Download Agent SHOULD allow users to remove Media Objects. When a Media Object is to be removed from the device, the user SHOULD be prompted to confirm that the Media Object may be removed. The Download Agent SHOULD warn the user of any special circumstances that arise during the deletion of the Media Object.

5.4.3 Local Content Presentation

The user experience can often be enhanced if the user interface is graphical rather than text based. The optional `iconURI` provides a facility to define an icon that can be used to represent the object on the screen of the device.

If present in the Download Descriptor the URI should contain a reference to a small graphical object. The means, including capability negotiation facilities, to download the graphical object are defined in the scheme of the element `iconURI`.

The support for `iconURI` is OPTIONAL in both the Download Descriptor as well as in the Download Agent. The Download Agent SHOULD retrieve and use the specified icon if this feature is supported.

5.4.4 Persistence of Download Descriptor elements

The Download Agent MAY use the elements received in the Download Descriptor in association with the Media Object at its discretion. Some of the elements MAY be stored persistently in conjunction with the installation of the Media Object.

If the Download Descriptor contains the following Download Descriptor elements and if the Download Agent support at least one of these elements, the Download Agent MUST persistently store the Download Descriptor associated with the Media Object to be able to use these elements later.

The Download Descriptor element that is used for sending deletion notification (see section 5.4.2) is as follows:

- `deleteNotifyURI`

The Download Descriptor elements and HTTP header that are used for updating Media Objects (see section 5.2.4.2) are as follows:

- `updatedDDURI`
- `objectID`
- `objectVersion`
- `Etag header`

For a Product that contains the abvove listed Download Descriptor elements and HTTP header listed above, the Download Agent MUST persistently store the elements and HTTP header above associated with the Download Descriptor, Product or Media Object to be able to use these elements and HTTP header later.

The Download Agent is free to store the elements above in any format and in any location, i.e. it is out of scope for this specification. If a Media Object is moved to a different device with a portable memory card or managed from an external entity, for example a PC, the elements above may be lost and it may be impossible to invoke the procedures associated with the elements above.

5.5 HTTP Specific Functionality

5.5.1 Client capability advertisement

For download operations over HTTP [RFC2616] or HTTPS [RFC2818] the device should advertise its capabilities (to the extent possible) by using the mechanism of HTTP request headers. The Server (or servers) supplying the Download Descriptor and the Media Object should use this information to select content that is appropriate for the device.

The Download Agent SHOULD include the following HTTP headers.

- Accept [RFC2616] (See Section 5.5.1.1)
- User-Agent [RFC2616] or UAProf [UAProf] (See Section 5.5.1.1).

5.5.1.1 Accept Header

If the Download Agent chooses to advertise its supported media types using the HTTP Accept header, then the Download Agent MUST at least advertise support of the media type of the Download Descriptor. The media type of the Download Descriptor is as follows:

- `application/vnd.oma.dd2+xml`

The support of the media type must also be exposed during the content discovery phase so that the Download Server can recognize that the device supports the Download Descriptor. Thus, the discovery application that supports the OMA Download OTA version 2 Download Agent must advertise the support of the media type. The Download Server MUST include the MIME type `application/vnd.oma.dd2+xml in the Content-type header when a Download Descriptor is included in a HTTP response.`

See Appendix F for more information about the registration of the media type.

5.5.1.2 User Agent Profile Headers

The Download Agent SHOULD advertise its capability using UAProf [UAProf] unless the Download Agent advertises its capability using User-Agent [RFC2616]. If the Download Agent decides to advertise its capability using UAProf, the Download Agent MUST advertise the attribute in the table below as indication in the “MUST Advertise” column. The support of UAProf is OPTIONAL for both Download Agents and Download Servers.

UAProf Attribute	Description	Resolution Rule	Type	Sample Values	MUST Advertised
OmaDownload	Sent “Yes” if the device supports Download OTA.	Locked	Boolean	“Yes”, “No”	The Download Agent MUST send the “Yes” or “No”.
OmaDownloadVersion	Supported Download OTA Enabler Release version in “<major>.<minor>” format supported by the Download Agent.	Append	Literal	“2.0”	The Download Agent MUST send the supported version. If the Download Agent supports multiple versions, it MAY send multiple supported versions.
OmaDownloadMoSupported	Number of supported Media Objects per Product. The Download Agent MUST support at least one MediaObject per Product. If the Download Agent supports the download of Multiple Media Objects grouped within a single Download Descriptor it MUST support at least 10 Media Objects per Product.	Locked	Literal	“3”, “1000”	The Download Agent MUST send the value if it supports multiple Media Objects per Product.
OmaDownloadProductsSupported	Number of supported Products per Download Descriptor. The Download Agent MUST support at least one Product per Download Descriptor.	Locked	Literal	“3”, “1000”	The Download Agent MUST send the value if it supports multiple Products per Download Descriptor.
OmaDownloadTimingReservationSupported	Indicates is the Device supports the Download Timing Reservation functionality	Locked	Boolean	“Yes”, “No”	The Download Agent MUST send the value if it supports the Download Timing Reservation.
OmaMediaObjectUpdate	Indicates is the Device supports the Media Object Update using objectID and objectVersion elements in the Download Descriptor	Locked	Boolean	“Yes”, “No”	The Download Agent MUST send the value if it supports Media Object Update using objectID and objectVersion elements in the Download Descriptor
OmaDownloadPauseResume	Sent “Yes” if the Download Agent supports the Pause and Resume Object Retrieval function as defined in 5.2.4.3.	Locked	Boolean	“Yes”, “No”	The Download Agent MUST send the attribute if it supports the Pause and Resume function.
OmaDownloadServerInitiatedDownload	Sent “Yes” if the Download Agent supports Server Initiated Automatic Download as	Locked	Boolean	“Yes”, “No”	The Download Agent MUST send the attribute if it supports the Server Initiated Automatic

UAProf Attribute	Description	Resolution Rule	Type	Sample Values	MUST Advertised
	defined in 5.6.				Download function.
OmaDownloadCompletion	Sent "Yes" if the Download Agent supports Download Completion Notification as defined in 5.2.6.	Locked	Boolean	"Yes", "No"	The Download Agent MUST send the attribute if it supports the Download Completion Notification function.
OmaDownloadDelation	Sent "Yes" if the Download Agent supports Deletion Notification as defined in 5.4.2.	Locked	Boolean	"Yes", "No"	The Download Agent MUST send the attribute if it supports the Deletion Notification function.
OMADownloadProgressiveDLAccept	List of content types that the device can render the Media Object as it is being downloaded.	Literal	Append	"video/3gpp", "audio/mpeg"	The Download Agent MUST send supported media types that are progressive downloadable.
OMADownloadEtagUpdate	Sent "Yes" if the Download Agent supports Media Object updated using Etag as defined in 5.2.4.2.	Locked	Boolean	"Yes", "No"	The Download Agent MUST send the attribute if it supports the Media Object update using Etag function.
OMADownloadLictype	List of `lictype` that the device can handle. The format SHOULD be a WAP application ID as defined in 7.2.3.2.8.	Literal	Append	"x-wap-application:drm.ua"	The Download Agent MUST send supported `lictype` if it supports processing the `license` element.
OMADownloadLicmimetype	List of `licmimetype` that the device can handle. The format MUST be a MIME media type as defined in 7.2.3.2.8.	Literal	Append	"application/vnd.oma.drm.roap-trigger+xml"	The Download Agent MAY send supported `licmimetype`.
OMADownloadEnvironment	List of `envtype` that the device can handle. The format MUST be a URI as defined in 7.2.3.2.10.	Literal	Append	"urn:oma:xml:dl:jad:2.0"	The Download Agent MUST send supported `envtype` it supports the `environment` element.

5.5.2 State Management of download transaction

State Management in the download transaction can be handled using multiple different methods. The definition of these methods is outside the scope of this specification except in case for server initiated download, see section 5.6. This section gives two examples of methods that MAY be supported by a Download Agent.

The state management is most relevant in a transaction that leverages the Status Report Functionality. In this case it is important for the server to be able to associate the offer to download a Media Object (the Download Descriptor), the actual retrieval of the Media Object, and the Status Reports.

The first method is based on the URL's that are included in the Download Descriptor. Each of the URL's may have a session identifier parameter that allows the server to associate the operations with each other.

The second method is based on the use of cookies, as defined in [HTTPSM]. This method allows the server to issue a cookie that will be associated with each subsequent operation within the download transaction.

5.5.3 Transparency of Download Descriptor mechanism

The Download Descriptor download transaction mechanism is designed to be transparent, i.e. from the content handler (i.e. GIF, JPEG, MIDI, etc.) point of view there should be no difference if the object was downloaded directly using a plain HTTP request-response, or using the Download Descriptor mechanism. If the content handler or the system in general, can benefit from information conveyed in the HTTP headers, then these headers should be available in a transparent manner.

The Download Descriptor transaction consists of several request-response pairs, all of them part of the transaction, and all of them with associated HTTP headers. The Download Agent MUST make the headers associated with the actual Media Object transfer available together with the Media Object. Headers associated with the other request-response operations SHOULD NOT be exposed to the Media Object specific content handler.

5.6 Server Initiated Automatic Download

OMA Download OTA v2 includes the ability for a Download Server to automatically trigger the download and installation of a Media Object. This is achieved by pushing the Download Descriptors to devices via push delivery mechanisms such as:

- WAP Push [WAPPush]
- MMS [MMS]

The Download Agent SHOULD support the Server Initiated Automatic Download function. If the Download Agent supports this function, then the Download Agent MUST at least support [WAPPush]. The Download Server SHOULD support the Server Initiated Automatic Download function.

The following Push Application ID MUST be used when using WAP Push [WAPPush] to deliver Download Descriptors to the Download Agent.

- Application ID URN: `x-oma-application:dlota.ua`
- Application ID binary number: 0x11 (This value was assigned by [OMNA].)

The Download Agent MUST be able to receive and process Download Descriptors that are pushed using the Push Application ID defined above.

Since there may be times when the Download Descriptor is too large to fit into one push message, e.g. if SMS is used for underling network bearer, the Download Server MAY use Service Indication [SI] or Service Loading [SL] to notify the Download Agent that a Download Descriptor is available. The Download Agent MUST support Service Indication [SI] and Service Loading [SL] if the Download Agent supports the Server Initiated Automatic Download function.

When a Download Agent receives a trigger for download, it MUST check if the originator of the trigger message is in the white list (Defined in Section 8.3) stored in the terminal. The Download Agent MUST only accept the trigger if the originator of the trigger is in the white list.

In the server initiated automatic download use case, a malicious party can transmit a large number of download triggers to a large set of Download Agents to start a download session at the same time. In order to protect and mitigate against Denial of Service (DoS) attacks against Download Servers, a Session ID MAY be included in `objectURI` as specified below.

The Download Server MAY include a randomly selected unique Session ID in the `objectURI` of the trigger that is used to initiate the request for the Download Descriptor. The Session ID MUST at least be 128bits long and randomly generated for each download trigger.

Request-URI of the Download Descriptor MUST contain an URI parameter "requesttype" that MUST be set to "register", i.e. Request-URI takes the form of "/bmsc.home1.net/keymanagement?requesttype= register

Upon receiving a trigger containing the `objectURI` for the Download Descriptor, the Download Agent MUST contact the Download Server and present the Session ID in the DD request message. The Request-URI of the Download Descriptor shall contain an URI parameter “sessionid” that shall be set to value of the Session ID received in the triggering message.

When the Download Server receives the request containing the Session ID, it MUST ensure that the Session ID corresponds to a valid Session ID that was pushed to the user as part of the URI for the Download Descriptor. In cases where the Session ID received does not match the Session ID sent the Download Server MUST reject the download request.

An example of the Request-URI is shown as follows:

```
/bmsc.home1.net/picture.dd?sessionid=TUlEbGV0LTE6TUlE
```

6. Status Report Functionality

Within OMA Download OTA version 2.0, there are four types of notifications. These notifications are used to provide feedback on the status of the download and installation operations. These notifications can be used by a service to determine how many Media Objects were downloaded and installed, or if a large number of users aborted the download when they were presented with a formal description of the Media Object.

The confirmation of a successful download and installation operation is particularly useful in deployments where some kind of pay-per-transaction business model is used. The Status Report can also be used to optimise the allocation of server resources.

However, it should be noted that a server cannot ever fully rely on the reception of a Status Report to indicate a completed transaction. The device may be unable to send the status report. Therefore the server needs to employ robust logic to discard hanging transactions.

6.1 DLOTAv2 Status Reports

The status reports supported by OMA Download OTA version 2.0 are followings:

- Reservation Related Notifications

Reservation Related Notifications indicate to the Status Report Server that there was a success to set the timer or an error to execute the download timing reservation as specified in Section 5.2.4.1. Table 1 shows the status codes that are sent to the URI that is designated by the `reservationNotiryURI` element as defined in section 7.2.5.4.

Status Code	Status Message	Informative description of Status Code usage
971	Set Timer Success	Indicates that the Download Agent was able to set the timer at the time designated by the `downloadTime` element.
972	Reservation Error	Indicates that the Download Agent was not able to execute the download transaction at the designated time,
973	Reservation Cancelled	Indicates that after receiving the Download Descriptor with timing reservation related elements, the Download Agent or User decides to cancel timing reservation.

Table 1: Status codes used for Reservation Related Notifications

- Download Completion Notification

The Download Completion Notification indicates to the Status Report Server whether the Download Agent successfully downloaded the Media Object or Product or not. The `downloadNotifyURI` MAY be used to report the detailed reason for failure, but only if `installNotifyURI` element is not present in the Download Descriptor. At the point where this notification is sent the Media Object or Product is not available to the user. Table 2 shows the status codes that are sent to the URI that is designated by the `downloadNotifyURI` element as defined in section 7.2.3.2.11. This notification can be used to achieve the pre-downloading Media Objects use case [DLREQ].

Status Code	Status Message	Informative description of Status Code usage
970	Mixed Status	Indicates that the Media Objects of a Product have different statuses. Additional lines will contain the status for each Media Object of the Product. In this case the additional lines SHALL only consist of a subset of the following status codes allowed for the Download Completion Notification as specified within this table.
981	Download Completion	Indicates that the Media Object is correctly downloaded to the Download Agent. In this state, the Media Object is yet to be available to the user.
982	Download Error	A generic status code to indicate to the service that the device could not successfully download the content. If the installNotifyURI is present in the Download Descriptor the full status report MUST contain the appropriate status code(s) from Table 3. If the installNotifyURI element is not present in the Download Descriptor the downloadNotifyURI MAY be used to send a full status report that provides the detailed reason(s) for failure. In this case the full status report MUST include the appropriate status code(s) from Table 3.

Table 2: Status codes used for Download Completion Notification

- Installation Notification

The Installation Notification indicates the installation status of the Media Object or Product to the Status Report Server. If the installation was successful the Media Object or Product will be available to the user after the Download Agent has completed its obligations to send the Installation Notification as specified in Section 5.3.2. In the case of a failure the Installation Notification indicates that the Download Agent encountered an error during the installation of the Media Object or Product. If installation fails then the Media Object or Product MUST be discarded except when the Download Agent will resume the download transaction as define in section 5.2.4.3. Table 3 shows the status codes that are sent to the URI that is designated by the `installNotifyURI` element as defined in section 7.2.3.2.7.

If the `installNotifyURI` is defined for a Product, the Download Agent SHALL send the notification as soon as all Media Objects are downloaded and prepared for installation. If the `installNotifyURI` is defined for a Product, it MUST NOT be defined for any of the Media Objects of the Product.

Status Code	Status Message	Informative description of Status Code usage
900	Success	Indicates to the service that the Media Object was downloaded and installed successfully.
901	Insufficient Memory	Indicates to the service that the device could not accept the Media Object as it did not have enough storage space on the device. This event may occur before or after the retrieval of the Media Object.
902	User Cancelled	Indicates that the user does not want to go through with the download operation. The event may occur after the analyses of the Download Descriptor, or instead of the sending of the Installation Notification (i.e. the user cancelled the download while the retrieval/installation of the Media Object was in process).
903	Loss of Service	Indicates that the client device lost network service while retrieving the Media Object.

905	Attribute Mismatch[1]	Indicates that the Media Object does not match the elements and attributes defined in the Download Descriptor, and that the device therefore rejects the Media Object.
906	Invalid Descriptor	Indicates that the device could not interpret the Download Descriptor. This typically means a syntactic error.
907	Invalid Type	Indicates that the device does not support one or more of the media types specified by the Type(s) element.
951	Invalid DDVersion	Indicates that the device was not compatible with the "major" version of the Download Descriptor, as indicated in the `DDVersion` attribute (that is a parameter to the `media` element).
952	Device Aborted	Indicates that the device interrupted, or cancelled, the retrieval of the Media Object despite the fact that the content should have been executable on the device. This is thus a different case from "Insufficient Memory" and "Non-Acceptable Content" (where the device has concluded that it cannot use the content).
953	Non-Acceptable Content	Indicates that after the retrieval of the Media Object, but before sending the installation notification, the Download Agent concluded that the device cannot use the Media Object.
954	Loader Error	Indicates that the URL that was to be used for the retrieval of the Media Object did not provide access to the Media Object. This may represent for example errors of type server down, incorrect URL and service errors.
955	Media Object Updated	Indicates that the Download cancelled the Media Object Download due to the fact the Media Object has been updated.
956	License Retrieval Success	Indicates that the License retrieval for the Product or Media Object succeeded. This error MUST be reported separately from other errors. See paragraph 6
957	License Retrieval Failed	Indicates that the License retrieval for the Product or Media Object Failed. This error MUST be reported separately from other errors. See paragraph 6
958	Version Already Available	Indicates that the version of the Media Object is already available on the device.
959	Envtype Not Supported	Indicates that the device does not support the `envtype` that was specified in `environment` element of the Download Descriptor
960	Environment Internal Status	Indicates that the status report includes the status according to the specific environment as returned by the environment specific content handler. A line shall follow with the internal status code (e.g. in the case of MIDP the installation status returned by the MIDP AMS).
970	`Mixed Status`	Indicates that the Media Objects of a Product have different statuses. Additional lines will contain the status for each Media Object of the Product.

Table 3: Status codes used for Installation Notification

- Deletion Complete Notification

The Deletion Completion Notification indicates to the Status Report Server that the Media Object or Product has been removed from the device, and the Media Object or Product is no longer available to the user. Table 4 shows the status codes that are sent to the URI that is designated by the `deleteNotifyURI` element as defined in section 7.2.3.2.12.

Status Code	Status Message	Informative description of Status Code usage

[1] To be compatible with [MIDP], this specification uses the term "Attribute Mismatch" as the status message.

991	Deletion Complete	Indicates that the Media Object was deleted from the device.

Table 4: Status codes used for Deletion Notification

6.2 Status Report Formatting

The Status Report MUST be formatted according to the ABNF syntax defined in this section. The ABNF notation used is defined in [RFC4234]. The terminals <DIGIT>, <WSP> and <CRLF> are also defined in [RFC4234]. The terminal <absoluteURI> is defined in [RFC3986].

```
fullStatusReport = overallStatusReport [ *(CRLF detailStatusReport) ] [CRLF environmentStatus] [ CRLF ]

overallStatusReport = overallDownloadStatus [ CRLF overallLicenseStatus ]
overallDownloadStatus = statusLine
overallLicenseStatus = statusLine

detailStatusReport = detailDownloadStatus [ CRLF detailLicenseStatus ]
detailDownloadStatus = detailStatusLine
detailLicenseStatus = detailStatusLine

statusLine = statusCode 1*WSP statusDescription
detailStatusLine = absoluteURI 1*WSP statusLine

statusCode = 3DIGIT
statusDescription = *(VCHAR / WSP)

environmentStatus=*(VCHAR/WSP/CRLF/3DIGIT)
```

A <fullStatusReport> always contains an <overallStatusReport> that describes the status for the Product or Media Object that contains the `installNotifyURI`. In case the <fullStatusReport> is the status of a Product, additional <detailStatusReport> elements MAY be included, one for each Media Object within the Product. The additional <detailStatusReport> elements, one for each Media Object of the Product, MUST be included in case the Media Objects of the Product do not have the same Status for downloading the Media Object, or in case any of the Media Objects of the Product has a `license` element.

If the Media Object or Product contains a `license` element then the <overallStatusReport> or <detailStatusReport> element defining the status of the Media Object or Product MUST contain an <overallLicenseStatus> or <detailLicenseStatus> respectively.

The <statusCode> report element MUST be a Status Code as defined in Table 1 to Table 4 in section 6.1. The <statusDescription> MUST be a Status Message as defined in Table 1 to Table 4 in section 6.1. The <statusCode> and <statusDescription> for a single <statusLine> must be obtained from the same row from Table 1 to Table 4 in section 5.2.

If the Media Object or Product contains an `environment` element then the <overallStatusReport> MAY contain a <environmentStatus> element. The <statusCode> reported for the <statusLine> within the <overallStatusReport> of a <fullStatusReport> that contains a <environmentStatus> element MUST be set to 960.

The <statusCode> reported for the <statusLine> within the <overallStatusReport> of a Product MUST be set to 970 in case the download Status Code of the Media Objects of the Product are not all equal.

Examples of the status report message is show in Appendix E.4.

7. Download Descriptor

The Download Descriptor is a collection of elements, used to describe a Media Object. The defined elements are specified to allow the Download Agent to identify, retrieve, and install Media Objects. It may also be used by the application that is actually processing the Media Object (the Content Handler); the Download Descriptor may contain media specific elements.

Section 7.2 defines the semantics of the Download Descriptor. The XML schema of the Download Descriptor is defined in [DDXSD].

7.1 Download Descriptor

The Download Descriptor is used by the Download Agent and by the content handler that ultimately processes the Media Object. The Download Descriptor may for example include content handler specific elements. The Download Agent SHOULD expose the complete Download Descriptor to the content handler (at the request of the content handler. The interface may be the same as for HTTP headers).

The descriptor allows the device to verify that the desired Media Object is suitable for the device before being loaded. It also allows Media Object-specific elements (parameters) to be supplied to the relevant content handler.

The client device MUST use the MIME media type declared by the transport or packaging mechanism to identify a Download Descriptor object. The MIME media type is defined in Section 7.4 XML Syntax for Download Descriptor.

A predefined set of elements is specified to allow the Download Agent software to identify, retrieve, and install Media Objects. All elements appearing in the Download Descriptor are made available to the content handler of the media type that the Download Descriptor references.

Basically, a Download Descriptor contains a single `media` element. This `media` element optionally consists of a single `vendor` element and one or more `product` elements. The `product` elements in turn contain multiple `mediaObject` elements. A more detailed description of this structure can be found in the next sections. See Section 7.4 for the exact XML syntax.

7.2 Download Descriptor elements and attributes

The elements and attributes in the descriptor MUST be formatted according to the syntax defined in the section 7.4. If not, then an error code “Invalid Descriptor” MUST be returned in the status report. However, it will in many cases be impossible to send the error code in case of a Download Descriptor that cannot be parsed properly due to formatting errors.

Descriptors retrieved via HTTP should use the standard HTTP content negotiation mechanisms, such as the Content-Encoding header and the Content-Type charset parameter to decode the stream to the preferred character set for the actual MIME media type representation of the Download Descriptor.

Each element and attribute is defined using the following properties:

Name - The name of the element

Definition - A statement that clearly represents the concept and essential nature of the element

Status - Whether the element is Mandatory – it MUST be included in a valid Download Descriptor - or Optional - MAY be included in the Download Descriptor. The property also defines if support for the functionality is optional or mandatory in the Download Agent.

Datatype - Indicates the type of data that can be represented in the value of the element

Refinement - A qualifier that makes the meaning of the element narrower or more specific

Comment - A remark concerning the application of the element

The elements and attributes are defined in the following sections.

7.2.1.1 DDVersion

Name	DDVersion
Definition	Attribute defining the version of the Download Descriptor.
Status	Download Descriptor: Mandatory. User Agent: Optional
Datatype	String (decimal)
Refinement	
Comment	The format of the DDVersion is "major.minor". A Download Agent not supporting the "major" version of the Download Descriptor version SHOULD send an "Invalid DDVersion" status code to the installNotifyURI and reject the Download Descriptor. The device MAY skip sending a status report if it is unable to parse the Download Descriptor The "minor" version is used to differentiate between backwards compatible versions of the Download Descriptor. The version of the Download Descriptor defined in this specification is "2.0". The default DDVersion, when the attribute is omitted from the Download Descriptor, is "1.0"

7.2.2 vendor

Name	vendor
Definition	Information over the organisation that provides the Media Object. It consists of a name, a home, a logo and a support element.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	The parent element of name, home, logo, support
Refinement	Does not have any particular semantics, is intended for user interpretation.
Comment	The element MAY be displayed to the user during installation. The element MAY be used by the Download Agent to create a unique name for the Media Object when it is stored in the device. The Download Agent MAY allow the user to follow any of the optionally define URI's for more information about the vendor.

7.2.2.1 name

Name	name
Definition	If defined as child element of mediaObject, a user readable name of the Media Object that identifies the object to the user. If defines as child element of vendor, a user readable name of the organisation providing the Media Objects.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	String
Refinement	Does not have any particular semantics, is intended for user interpretation.
Comment	The client MAY use the name as the default storage name, or as a part of the

	storage name if it is defined as a child element of the `mediaObject`. The Download Agent MAY also use the `name` element of the vendor to ensure uniqueness between names. The client MAY additionally use the `text` elements of the `meta` element for generating storage names.

7.2.2.2 home

Name	`home`
Definition	The optional URI pointing to the Home page of the Vendor.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-.
Comment	The Download Agent MAY allow the user to browse to the URI to obtain additional information about the vendor.

7.2.2.3 logo

Name	`logo`
Definition	The optional URI pointing to a vendor specific image.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-.
Comment	The Download Agent MAY use this URI to obtain the logo of the vendor and present it to the user.

7.2.2.4 support

Name	`support`
Definition	The optional URI pointing to the Support page of the Vendor.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-.
Comment	The Download Agent MAY allow the user to browse to the URI to obtain vendor specific support information. The vendor could present a context specific page to the user, helping the user to recover from a failed download.

7.2.3 product

Name	`product`
Definition	Container of one or more Media Objects.
Status	Download Descriptor: Mandatory. User Agent: Mandatory

Datatype	The parent element of `meta` and `mediaObject`
Refinement	The Product can be marked as a Compound Product by adding a `compound` attribute. When the Product is defined to be a Compound Product, the Download Agent MUST handle the contained Media Objects as if it is a single Media Object during the capabilities check, or when sending install, download or delete notification. For a Compound Product notifications MUST not be defined for any Media Object of the Product, only for the Product as a whole.
Comment	This element wraps the media objects and generic meta information about the Product The Download Agent MUST accept at least one `product` element in a single Download Descriptor.

7.2.3.1 compound

Name	`compound`
Definition	Attribute for a Product that defines the Product to be a Compound Product.
Status	Download Descriptor: Optionally. User Agent: Mandatory
Datatype	Boolean
Refinement	True means that the Product is a Compound Product. Additional processing rules exist for such Product. False means that the Product is just a collection of Media Objects. The Download Agent MAY however treat the Media Objects as separate entities. If not defined, the default value is false.
Comment	-

7.2.3.2 meta

Name	`meta`
Definition	Container of Media Object or Product specific meta information.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	The parent element of `name`, `description`, `text`, `infoURL`, `iconURI`, `installParam`, `installNotifyURI`, `license`, `environment`, `downloadNotifyURI`, `deleteNotifyURI` and `suppressUserConfirmation`.
Refinement	The `meta` element contains several of the other defined elements that are both valid for Product and Media Object. The semantics of the elements in the `meta` element differ slightly as defined below: For elements, except the `license` element or notification elements: The Download Agent SHOULD use the value as defined for the Media Object. If an element is not defined for the Media Object, but is defined for the Product, the Download Agent SHOULD use the `product` elements as if they are also defined for the Media Object. For notification elements: If defined for a Product, the notification elements MUST

	NOT be defined for any of the Media Objects within the Product. Notifications should be done when all Media Objects of the Product have the same state. If a notification element is defined at product level, then the same notification element if defined for Media Objects within the Product SHOULD be ignored by the Download Agent for such Media Objects. For `license` element: When defined for a Media Object or Product, the Download Agent MUST pass the contents of the `license` element to the appropriate License Agent. The element can be defined for the Product and for zero or more of the Media Objects of the Product. In this case, the Download Agent MUST pass all the contents of the `license` elements to the appropriate License Agent as separate requests. The suppressUserConfirmation element can be defined at both the Product level and the Media Object level, see section 5.2.3 for details.
Comment	-

7.2.3.2.1 name

Name is defined in section 7.2.2.1

7.2.3.2.2 description

Name	`description`
Definition	A short textual description of the Media Object
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	String
Refinement	Does not have any particular semantics, is intended for user interpretation.
Comment	The `description` element SHOULD be displayed to the user before the download of the Media Object is accepted by the end user. The `description` element allows the user a last check before the transaction is completed.

7.2.3.2.3 text

Name	`text`
Definition	Container that defines additional Media Object type specific textual meta data.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	An element to include `id` and `display` attributes
Refinement	This element contains textual information that does not have any particular semantics, and is intended for user interpretation The element has two xml attributes that provide information about the semantics (`id`), and a user readable string defining the semantics (`display`).
Comment	The Download Agent MAY use the information for generating storage names. The Download Agent SHOULD present the contents and the semantics string to the user when providing information to the user about a Product or Media Object.

7.2.3.2.4 infoURL

Name	infoURL
Definition	A URL for further describing the Media Object
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URL [RFC3986]
Refinement	Does not have any particular semantics, is intended for user interpretation.
Comment	The infoURL can be used to retrieve information that describes the Media Object.

7.2.3.2.5 iconURI

Name	iconURI
Definition	The URI of an icon
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-
Comment	The iconURI may be used by the client to represent the Media Object (e.g. an application) in the user interface (e.g. application manager).

7.2.3.2.6 installParam

Name	installParam
Definition	An installation parameter associated with the downloaded Media Object
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	Opaque text string
Refinement	-
Comment	The value is an opaque text string that is handed by the Download Agent to the Media Object Installer. The syntax and semantics of the opaque string is relevant only to the particular Media Object Installer. The value is fully transparent to the Download Agent.

7.2.3.2.7 installNotifyURI

Name	installNotifyURI
Definition	The URI (or usually URL) to which an Installation Status Report is to be sent, either in case of a successful completion of the download, or in case of a failure.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	URI [RFC3986]
Refinement	-
Comment	If the installNotifyURI element is defined then the Download Agent MUST

	send an Installation Status Report both in the case of success and any kind of failure. The status code is as defined in Table 3. If the element is missing then no Installation Status Report can be issued, neither for success nor for failure. The Download Agent posts a status-report to this URI. The URI is used both to report errors and process aborts, as well as to verify the successful installation of the Media Object.

7.2.3.2.8 license

Name	`license`
Definition	Information for a License Agent to retrieve a License for the Product or Media Object.
Status	Download Descriptor: Optional. User Agent: Optional (see below)
Datatype	XML schema *any* and its XML attribute `lictype` is an URI [RFC3986] and its XML attribute `licmediatype` is a MIME media type of the embedded license.
Refinement	The `order` attribute can be used to define that the license information shall be passed to the License Agent after the Media Object or Product has been downloaded.
Comment	This element wraps the License specific meta information described by namespace qualified XML elements. The `lictype` value unambiguously identifies the information set that can be included inside the `license` element and the License Agent that is able to retrieve the License for the Media Object or Product. If the License Agent is unknown then the Download Agent SHOULD abort the download transaction. The possible values of `lictype` SHOULD be a WAP Push application ID as registered by [OMNA]. The `licmediatype` value identifies the MIME media type that can be included inside the `license` element. If the MIME media type is unknown then the Download Agent SHOULD abort the download transaction. The values of `mimelictype` MUST be a MIME media type as defined in [RFC2046]. If the Device supports OMA DRM v2.0 then the Device MUST support the `licence` element and the processing of the `licence` element as defined in sections 5.3.2 and 5.2.5. If the Device does not support OMA DRM v2.0 the Device MAY support the `license` element.

7.2.3.2.9 order

Name	`order`
Definition	Attribute for a `license` that defines whether Download Agent is free to choose when the contents of the license element shall be passed to the License Agent or whether the contents of the license element shall be passed to the License Agent after the Media Object or Product has been Downloaded.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	String
Refinement	`any` means that the Download Agent is free to chose when to pass the License information to the License Agent.

	`post` means that the License information MUST not be passed to the License Agent until the associated Media Object or Product has been completely Downloaded and Installed.
Comment	-

7.2.3.2.10 environment

Name	`environment`
Definition	Container of execution environment specific metadata needed for the Media Object processing.
Status	Download Descriptor: Optional. User Agent: Optional (see below)
Datatype	XML schema *any* and its XML attribute "envtype" is an URI [RFC3986]
Refinement	
Comment	This element wraps the environment specific meta information described by namespace qualified XML elements. The `envtype` value unambiguously identifies the information set that can be included inside the `environment` element and the content handler of the Media Object. If the content handler is unknown then the client SHOULD abort the download transaction and post the "Envtype Not Supported" (959) status report if the `installNotifyURI` element is present in the Download Descriptor. The possible values of `envtype`, syntax and semantics of the internal meta data structures depend on separate environment specific standards. If the device supports MIDP content then the device MUST support the `environment` element and the processing of the `environment` element as defined in section 10. If the device does not support MIDP the device MAY support the `environment` element.

7.2.3.2.11 downloadNotifyURI

Name	`downloadNotifyURI`
Definition	The URI (usually URL) to which a completion of download is to be sent.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Server MAY include the `downloadNotifyURI` element into the Download Descriptor if it wants to receive the Download Completion Status Report from the Download Agent. If the element is defined then the Download Agent MAY send a Download Completion Status Report in the case of download completion. The status code is as defined in Table 2 (Section 5.3 Object Installation Process). If the element is missing then Download Completion Status Report SHALL NOT be issued.

7.2.3.2.12 deleteNotifyURI

Name	`deleteNotifyURI`
Definition	The URI (usually URL) to which a delete completion is to be sent.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Server MAY include the `deleteNotifyURI` element into the Download Descriptor if it wants to receive the Deletion Complete status report from the Download Agent. If the element is defined then the Download Agent MAY send the Deletion Complete status report in the case where the Media Object is deleted. The status code is as defined in Table 4 (5.3 Object Installation Process). If the element is missing then Deletion Complete Status Report SHALL NOT be issued.

7.2.3.3 mediaObject

Name	`mediaObject`
Definition	Container of Media Object specific information to retrieve the Media Object.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	The Parent element of `meta`, `size`, `installSize`, `type`, `objectID`, `objectVesion`, `progressiveDownloadFlag` and `objectURI`
Refinement	Download Descriptors will normally contain at least one `mediaObject` element, however it is possible for a Download Descriptor to contain no `mediaObject` elements. This is allowed for the purposes of transporting DRM specific licence information to facilitate the retrieval of licenses when the content is already available on the device. If the Download Descriptor does not contain any `mediaObject` elements then the Download Descriptor MUST contain a `license` element specified within the `meta` element of the `product` element. If a Download Agent receives a Download Descriptor that does not contain any `mediaObject` elements and no `license` element specified within the `meta` element of the `product` element the Download Agent MUST discard the Download Descriptor and if an `installNotifyURI` element is present in the Download Descriptor the Download Agent MUST post a 906 “Invalid Descriptor” status report.
Comment	This element wraps the media object specific information to retrieve the media object.

7.2.3.3.1 meta

Meta is defined in section 7.2.3.1

7.2.3.3.2 size

Name	`size`
Definition	The number of bytes to be downloaded from the URI. i.e. the Download Size of the Media Object.
Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	Non Negative Integer
Refinement	-
Comment	The `size` element defines the Download Size for the Media Object and can be used to allocate sufficiently large data buffers for downloading the Media Object. The storage size may be different from the Download Size, if for example, compression or some packaging format is used. The `size` equals to zero "0" may be used if the size of the Media Object is unknown when the Download Descriptor is created. This may be happened when download reservation is taken place.

7.2.3.3.3 installSize

Name	`installSize`
Definition	The number of bytes of Unallocated Memory that is required in order to download and install the Media Object. The value of these elements represents the total amount of Unallocated Memory required to download and install the Media Object.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	Positive integer
Refinement	-
Comment	If present the `installSize` element specifies the number of bytes of Unallocated Memory that that is required for successful download and installation of the Media Object. This element can be used by the Download Agent to determine if it has sufficient storage space to download and install the Media Object. If present and if no additional memory is required to install the Media Object then the value of this element MUST be equal to the `size` element.

7.2.3.3.4 type

Name	`type`
Definition	The MIME media type of the Media Object
Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	MIME Media type
Refinement	-
Comment	The `type` element indicates the media type of the object to be executed or rendered. The `type` element may occur multiple times in case the client device needs to support multiple media types in order to process a composite object or a packaging mechanism. The value of the `type` element MAY be different from the

	media type indicated in the HTTP header “content-type” as transport packaging MAY be used. The `type` element(s) SHOULD be used by the client to evaluate its capabilities relative to the content to be downloaded. The `type` element is used to indicate to the client if the Media Object to be downloaded has a media type that is supported by the client. If the `type` is not supported then the client SHOULD abort the download transaction. The device MUST support multiple occurrences of the `type` element per Media Object in the Download Descriptor.

7.2.3.3.5 objectID

Name	`objectID`
Definition	The identification of the media object
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Agent MUST use the object ID to identify the object. The `objectID` field MUST contain a globally unique identifier for this Media Object. The value MUST be encoded using US-ASCII encoding. The value MUST be a unique URI according to [RFC3986]. The use of globally unique object ID is required for OMA DLOTAv2 and it is the responsibility of the content author to guarantee the uniqueness of the `objectID` within their own namespace. This element MUST be present when the Download Descriptor contains an `objectURI` element. This element MUST be present along with `objectVersion` element if the Download Server wants to enable updating Media Object functionality.

7.2.3.3.6 objectVersion

Name	`objectVersion`
Definition	The version of the Object
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	String (decimal)
Refinement	-
Comment	The format of the `objectVersion` is “major.minor”. The `objectVersion` element shall be incremented if the corresponding media object is updated. This element MUST be present along with `objectID` element if the Download Agent wants to enable updating Media Object functionality.

7.2.3.3.7 progressiveDownloadFlag

Name	`progressiveDownloadFlag`
Definition	A flag that indicates if the Media Object is to be rendered as it is being downloaded.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	Boolean
Refinement	"true" means that the client may render the Media Object as it is being downloaded, if the client has this capability "false" means that the client is to download and then render the Media Object some time later. The Media Object is not to be reproduced as it is being downloaded. If the Media Object of the Product containing the Media Object has a license element, the Download Agent MUST NOT make the Media Object(s) available for rendering before the License has been successfully installed..
Comment	If this element is missing, the Media Object is not to be reproduced as it is being downloaded.

7.2.3.3.8 suppressUserConfirmation

Name	`suppressUserConfirmation`
Definition	Used to indicate that the Download Agent should not request user confirmation.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	String Enurmeration: "Always", "UserConfirmStepOnly", "Never"
Refinement	The suppressUserConfirmation element can be defined at both the Product level and the Media Object level, see section 5.2.3 for details.
Comment	The Download Server MAY include the `suppressUserConfirmation` element in Download Descriptor. When this is present and the value is 'Always' AND the Download Server can be authorized (see section 8.3) then the Download Agent MUST NOT request user confirmation at any point in the download process. When this is present and the value is 'UserConfirmStepOnly' AND the Download Server can be authorized (see section 8.3) then the Download Agent MUST NOT request user confirmation prior to downloading the Media Object i.e. step 4 (section 5.2.3) should be skipped. For all other appropriate points in the download process the Download Agent may require user confirmation, see sections (5.1 to 5.4) for details. When this is present and the value is 'Never' then the Download Agent MUST request user confirmation prior to downloading the Media Object and may have to during other appropriate points in the download process, see sections (5.1 to 5.4) for details. When this element is not present the Download Agent MUST treat the Download Descriptor as if the `suppressUserConfirmation` is present and equal to 'Never'.

7.2.3.3.9 objectURI

Name	`objectURI`
Definition	The `objectURI` contains one or many `server` elements that define the set of URI's that can be used to Download the Media Object. The Download Agent can choose any of the URI's
Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	The parent element of `server`.
Refinement	-
Comment	-

7.2.3.3.10 server

Name	`server`
Definition	The URI (usually URL) from which the Media Object can be loaded.
Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Agent MUST be able to use an HTTP GET to reference this URI in order to retrieve the Media Object.

7.2.3.3.11 objectValidityTime

Name	`objectValidityTime`
Definition	The time limit indicates whether the Media Object is volatile, i.e. has an expiration date.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	dateTime Values of type datetime MUST conform to a single lexical representation defined in in section 3.2.7 of [XMLSchema]. This lexical representation is the extended format CCYY-MM-DDThh:mm:ssZ where CC denotes the century, YY denotes the year, MM denotes the month, DD denotes the day, T is the date/time separator, hh, mm, ss represent the hour, minute, and second respectively, and Z is the mandatory UTC indicator. For example, 2002-12-31T23:59:59Z represents December 31st, 2002, 23:59:59 UTC.
Refinement	-
Comment	If this attribute is missing, the Media Object is considered as persistent. If it is present the Media Object is said volatile. The Download Agent MAY inform the user that some Media Objects are obsolete and the user can choose whether the Media Object shall be deleted or not. Deletion of obsolete Media Objects can be triggered by several criteria such as: - Available memory is less than a critical threshold, e.g 5 %.

	- Periodic garbage collection. - Validity time has expired. It is out of scope for this specification to define the events that can trigger automatic deletion of obsolete Media Objects. Complementary behaviours: - If the user attempts to delete the Media Object at a time prior to validity expiration time, the Media Object MUST be deleted. - If `suppressUserConfirmation` is present and the value is equal to 'Always' or 'UserConfirmStepOnly' and the Download Server can be authorized, then the Download Agent SHOULD NOT request confirmation from the user in order to proceed with the deletion of the Media Object. - If `deleteNotifyURI` is present then the Download Agent MAY inform the server of the deletion of the Media Object.

7.2.4 updatedDDURI

Name	`updatedDDURI`
Definition	The URI to which the Download Agent MUST retrieve the Download Descriptor of updated Media Object(s).
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Agent MUST be able to use an HTTP GET to reference this URI in order to retrieve the Download Descriptor of updated Media Objects.

7.2.5 reservation

Name	`reservation`
Definition	Inlcudes all reservation parameters that defines when content shall be downloaded.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	The parent element of `timestamp, download time, interval` and `reservationNotifyURI`.
Refinement	-
Comment	The `reservation` element is a wrapper for all parameters that define when a Media Object or Product shall be downloaded and optionally with which interval the Media Object shall be downloaded.

7.2.5.1 timestamp

Name	`timestamp`
Definition	The time measured by the Download Server.

Status	Download Descriptor: Optional. User Agent: Optional
Datatype	dateTime
Refinement	-
Comment	Values of type datetime MUST conform to a single lexical representation defined in section 3.2.7 of [XMLSchema]. This lexical representation is the extended format CCYY-MM-DDThh:mm:ssZ where CC denotes the century, YY denotes the year, MM denotes the month, DD denotes the day, T is the date/time separator, hh, mm, ss represent the hour, minute, and second respectively, and Z is the mandatory UTC indicator. For example, 2002-12-31T23:59:59Z represents December 31st, 2002, 23:59:59 UTC.

7.2.5.2 downloadTime

Name	`downloadTime`
Definition	The time for the automatic download.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	dateTime
Refinement	-
Comment	Values of type datetime MUST conform to a single lexical representation defined in section 3.2.7 of [XMLSchema]. This lexical representation is the extended format CCYY-MM-DDThh:mm:ssZ where CC denotes the century, YY denotes the year, MM denotes the month, DD denotes the day, T is the date/time separator, hh, mm, ss represent the hour, minute, and second respectively, and Z is the mandatory UTC indicator. For example, 2002-12-31T23:59:59Z represents December 31st, 2002, 23:59:59 UTC.

7.2.5.3 interval

Name	`interval`
Definition	Container of time interval related information
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	The parent element of `timeInterval` and `timeIntervalExpire`
Refinement	-
Comment	This element wraps the time interval related information for download reservation functionality. This element contains `timeInterval`, `timeIntervalExpire` elements.

7.2.5.3.1 timeInterval

Name	`timeInterval`
Definition	The time interval for planned download
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	Duration

Refinement	Values of type duration MUST conform to a representation defined in section 3.2.6 of [XMLSchema]. The lexical representation for duration is the extended format Pn MnDTnHnM, where nM represents the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, and nM the number of minutes. The number of years and seconds SHALL not be used. A preceding minus sign ('-') is not allowed. For example, P12H represents 12 hours interval.
Comment	The Download Server MAY include `timeInterval` element into Download Descriptor if the Download Server wants to execute the download transaction with the desired time interval. The Download Server SHALL include `downloadTime` element if `timeInterval` element is presented in the Download Descriptor. The Download Agent SHOULD execute the download transaction at or after the desired time indicated by `downloadTime` and `timeInterval` element. The desired time can be caluculated by `downloadTime` + n x `timeInterval` (n = 0, 1, 2, …). Adding duration to dateTime is specified in section E of [XMLSchema].

7.2.5.3.2 timeIntervalExpire

Name	`timeIntervalExpire`
Definition	The time limit for planned download.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	dateTime
Refinement	Values of type datetime MUST conform to a single lexical representation defined in section 3.2.7 of [XMLSchema]. This lexical representation is the extended format CCYY-MM-DDThh:mm:ssZ where CC denotes the century, YY denotes the year, MM denotes the month, DD denotes the day, T is the date/time separator, hh, mm, ss represent the hour, minute, and second respectively, and Z is the mandatory UTC indicator. For example, 2002-12-31T23:59:59Z represents December 31st, 2002, 23:59:59 UTC.
Comment	The Download Server MAY include `timeIntervalExpire` element into Download Descriptor if the Download Server wants to execute the download transaction at the desired time interval. The Download Server MUST include `timeIntervalExpire` element if `timeInterval` element is presented in the Download Descriptor. The Download Agent SHALL not execute the download transaction at or after the time indicated by `timeIntervalExpire` element.

7.2.5.4 reservationNotifyURI

Name	`reservationNotifyURI`
Definition	A URL to which a reservation status report is to be sent.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI [RFC3986]
Refinement	-
Comment	The Download Server MAY include `reservationNotifyURI` element into

	Download Descriptor if the Download Server wants to receive the Reservation Related Notifications as defined in section 6.1.

7.2.6 nextURL

Name	`nextURL`
Definition	The URL to which the client should navigate in case the end user selects to invoke a browsing action after the download transaction has completed with either a success or a failure.
Status	Download Descriptor: Optional. User Agent: Mandatory
Datatype	URL [RFC3986]
Refinement	-
Comment	`nextURL` provides a way for the download service to express the desired terminal behaviour in scenarios where the service to user interaction is to continue with browsing actions. This feature MAY for example be used when the Discovery Application is a Web browser.

7.3 Extensibility

The Download Descriptor used by OMA contains general metadata that is useful for all types of media. In some cases, however, the standardised elements are not sufficient and media type specific metadata must be added.

Extensions can be made to the Download Descriptor by defining the extension data in a separate namespace. That way, extension names will not collide with the standard metadata. The extensions can be used to trigger additional steps in the downloading procedure. The `environment` element MAY be used to wrap the environment specific meta information described by namespace qualified XML elements.

The extensibility is governed by a few basic rules:

- If an element is unknown to the Download Agent the element MUST be discarded
- If a value of an element is unknown to the Download Agent the element MUST be discarded

7.3.1 Media type with custom installation commands

The mechanism defined for the Download Descriptor is extensible. It is for example possible to extend the file with elements that are specific to the installation of a specific media type. However, it is recommended that the `installParam` element be used for custom installation commands as described in section 5.3.1.1, Media Object installation parameter.

The content handler for the Download Descriptor SHOULD evaluate its capabilities to download the object, and abort the download process in case it cannot complete it properly.

The content handler SHOULD evaluate the received Media Object (without indication to the user) before sending an Installation Notification indicating success. If it determines that it cannot process the received Media Object then the error code “Non-Acceptable Content” should be posted to the server, and the Media Object should be discarded.

7.4 XML Syntax for Download Descriptor

This section describes the syntax of the DD (Download Descriptor) media type. The syntax is expressed as XML [XML].

The media type `application/vnd.oma.dd2+xml` has been registered with IANA for use as the Download Descriptor (See Appendix F).

The Download Descriptor is defined using XML Namespaces [XMLNS]. The Download Agent MAY implement a fully namespaces aware XML processor as defined by [XMLNS], but is not required to do so in order to correctly process Download Descriptors.

The XML schema of the Download Descriptor is defined in [DDXSD].

8. DLOTAv2 Security

8.1 Download Agent Authentication

HTTP Digest [RFC2617] MUST be supported for Download Agent authentication by both the Download Server and the Download Agent. The use of Download Agent authentication MUST be enforced by the Download Server. If authentication is required, the Download Server MUST initiate authentication with in a HTTP session by sending a HTTP 401 (Unauthorized) message as specified in the HTTP Digest Specification [RFC2617].

In cases where HTTP Authentication Proxies are used to perform Download Agent authentication, the authentication proxy MUST indicate to the Download Agent inside an HTTP session that it requires authentication by sending a HTTP 407 (Proxy Authentication Required) as defined in the HTTP Digest Authentication specification [RFC2617].

For 3GPP implementations of Download Agents, HTTP Digest authentication SHOULD be performed according to 3GPP GAA [GAA] specifications.

In cases where 3GPP GAA is not supported, the Download Agent SHOULD support username/password provisioning mechanisms.

8.2 Server Authentication

The Download Server and the Download Agent MUST support TLS [RFC2246] server authentication to authenticate the Download Server. The need for Server Authentication MUST be indicated using the HTTPS prefix by the `objectURI`. HTTPS implementations MUST conform to RFC2818 [RFC2818]. Server Certificates used by the Download Server MUST conform to WAP Certificate profile [WAPCert].

8.3 Server Authorization

White lists SHALL be supported by the Download Agent to provide authorization of Download Servers. The address of each authorized Download Server MUST be configured to the Download Agent. OMA Device Management version v1.2 [OMADM] SHOULD be supported to enable the provisioning of the white list. If there are no server addresses configured in the white list then the Download Agent MUST always require user confirmation. Specifically for cases where the `suppressUserConfirmation` element is present in the Download Descriptor and the value is equal to ‘Always’ or ‘UserConfirmStepOnly’ and there are no server addresses configured in the white list, the Download Agent MUST NOT continue with the download process without acquiring user consent.

8.4 Confidentiality and Integrity Protection

When TLS is used to provide Download Server Authentication, the authenticated download MUST also be confidentiality and integrity protected using mechanisms defined in the TLS specification [RFC2246]. TLS implementations MUST conform to WAP Profile of TLS [WAPTLS] with the following additions:

The Download Server MUST implement the following cipher suites:

- TLS_RSA_WITH_AES_128_CBC_SHA [RFC3268].
- TLS_RSA_WITH_3DES_EDE_CBC_SHA.

The Download Agent MUST implement the following cipher suites:

- TLS_RSA_WITH_NULL_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA.

For Download Agent implementations that prefer additional cipher suites the Download Agent SHOULD implement:

- TLS_RSA_WITH_AES_128_CBC_SHA [RFC3268].

In cases where TLS is not used confidentiality and integrity protection SHOULD be provided by the underlying transport mechanism

9. Download OTA over Broadcast Bearer (Informative)

OMA Download OTAv2 permits to download Media Objects through a broadcast bearer. However, this specification does not specify technical details for this use case. It is outside the scope of this specification. However, the download of Media Objects can be achieved by relying on other broadcast data delivery methods such as 3GPP MBMS [MBMS], 3GPP2 BCMCS [BCMCS] and DVB-H [DVB-H].

One possible example of using Download OTA is to initiate the broadcast session by using the Download Descriptor to carry the necessary information. For example, the Download Descriptor is used to include a link to metadata such as metadata envelope objects and metadata fragment objects as defined by 3GPP MBMS [MBMS]. The metadata is downloaded by using OMA Download OTAv2 as the Media Object. Then the Download Agent provides the metadata to the broadcast application and then the broadcast application participates in or initiates the broadcast session.

10.Relationship to Java™ MIDP OTA

OMA Download OTA version 2.0 may be used for all downloaded content types including Java™ MIDlets. However, MIDP over the air specification [MIDPOTA] and the [MIDP20] specifications defines the mechanisms for downloading Java™ MIDlets. The Java™ Application Descriptor (JAD, as defined in [MIDPOTA]) is used for Java™ MIDlet downloads.

OMA Download version 2.0 provides the `environment` element that can wrap environment specific meta information described by namespace qualified XML elements. Therefore, the `environment` element can be used to wrap a JAD file so that the JAD file can be downloaded together with the Download Descriptor, thereby allowing service providers to support MIDlet download whilst still leveraging the advanced features that OMA Download OTA v2.0 provides.

This section defines how OMA Download OTA version 2.0 may be used to download MIDP MIDlets using the environment element. As specified in section 7.2.3.2.10 the `envtype` attribute of the `environment` element unambiguously identifies the information set that can be included inside the environment element and the content handler of the Media Object. For MIDP this envtype is:

```
urn:oma:xml:dl:jad:2.0
```

The corresponding schema to allow a MIDP JAD to be embedded within the `environment` element is defined in [JADXSD]:

If the device supports MIDP content then the device MUST support the `environment` element and the processing of the `environment` element as defined below. If the device supports MIDP content then in order to support superdistribution of MIDlets the AMS MUST supports the MIDlet Message format and the associated procedures as specified within [MIDPIMPBP].

If the device does not support MIDP content the device MAY support the `environment` element.

If the Download Descriptor contains the `environment` element and the `envtype` attribute of the `environment` element is not equal to any supported envtype and the Download Descriptor contains an `installNotifyURI` element then the Download Agent MUST post the "Envtype Not Supported" (959) status report, and abort the download transaction in the same manner that it would for other errors (See steps 7 and 9).

As there is overlap between the semantics of some of the Download OTA version 2.0 elements and some of the MIDP descriptor (JAD) attributes the following table defines the rules for these elements and attribute in Download Descriptors and JAD's that are wrapped within the `environment` element.

Download OTA v2 Element	MIDP Attribute	Rule
`Name`	MIDlet-Name	If the `name` element is present within the DD and MIDlet-Name attribute is present in the JAD. The value of the `name` element SHOULD be equal to value of the MIDlet-Name attribute.
`objectVersion`	MIDlet-Version	The `objectVersion` element represents the version of the Media Object version whilst the MIDlet-Version represents the version of the MIDlet. If the `objectVersion` element is present within the DD and MIDlet-Version attribute is present in the JAD the value of the `objectVersion` element MUST be equal to value of the MIDlet-Version attribute."
`Vendor`	MIDlet-Vendor	The `vendor` element MUST be present within the DD and the MIDlet-Vendor attribute MUST be present in the JAD the value of the `vendor` element MUST be equal to value of the MIDlet-Vendor attribute.

Download OTA v2 Element	MIDP Attribute	Rule
Description	MIDlet-Description	If the description element is present within the DD and the MIDlet-Description attribute is present in the JAD the value of the description element MUST be equal to value of the MIDlet-Description attribute.
infoURL	MIDlet-Info-URL	If the infoURL element is present within the DD and MIDlet-Info-URL attribute is present in the JAD the value of the infoURL element MUST be equal to value of the MIDlet-Info-URL attribute.
iconURI	MIDlet-Icon	The iconURI element MUST NOT be present within a DD that wraps a JAD using the environment element.
installNotifyURI	MIDlet-Install-Notify	The MIDlet-Install-Notify attribute MUST NOT be present within a JAD that is wrapped using the environment element. If the MIDP AMS receives a JAD from the Download Agent that was wrapped in the environment element and it includes the MIDlet-Install-Notify attribute the MIDP AMS MUST ignore this attribute and MUST NOT post an installation notification of its own, it MUST instead pass the status report to the Download Agent.
deleteNotifyURI	MIDlet-Delete-Notify	The deleteNotifyURI element MUST NOT be present within a DD that wraps a JAD using the environment element. If the Download Agent receives a DD that contains a JAD wrapped using the environment element and a deleteNotifyURI element the Download Agent must ignore the deleteNotifyURI element and treat the DD as if does not contain a deleteNotifyURI element. The JAD MAY contain the MIDlet-Delete-Notify attribute.
size	MIDlet-Jar-Size; Oma-Drm-Package-Size	The size element MUST be present in the DD and MIDlet-Jar-Size MUST be present in the JAD and the value of the size element MUST be equal to the MIDlet-Jar-Size. However, if the MIDlet is protected and the Oma-Drm-Package-Size attribute (as defined in [MIDPIMPBP]) is present in the JAD the value of the size element MUST be equal to the value of the Oma-DRM-Package-Size attribute and the value of the size element MAY be different to the value of the MIDlet-Jar-Size attribute.
objectURI	MIDlet-Jar-URL	The objectURI element MUST be present in the DD and the MIDlet-Jar-URL MUST be present in the JAD. For combined delivery of the DD and MIDlet (or DCF containing a MIDlet) the value of the objectURI element and the MIDlet-Jar-URL attribute MAY be different, for all other cases the value of the objectURI element MUST be equal to the value of the MIDlet-Jar-URL
downloadNotifyURI	Oma-Drm-Delivery-Notify	The Oma-Drm-Delivery-Notify attribute as defined in [MIDPIMPBP] MUST NOT be present within a JAD that is wrapped using the environment element. If the MIDP AMS receives a JAD from the Download Agent that was wrapped in the environment and it includes the Oma-Drm-Delivery-Notify attribute the MIDP AMS MUST ignore this attribute and MUST NOT post a delivery notification of its own, it MUST instead pass the status report to the Download Agent. If the DD contains the downloadNotifyURI element and the Download Agent supports the downloadNotifyURI element and its associated behaviour the Download Agent MUST post the appropriate status report to the downloadNotifyURI.

It should be noted that if the above rules are not followed and the relevant elements in the Download Descriptor and the corresponding attributes in the JAD contain different values when they should be equal then the behaviour is undefined.

Upon reception of a Download Descriptor containing the `environment` element with the `envtype` attribute of the `environment` element equal to the MIDP 2.0 envtype specified above:

- The Download Agent MUST treat the download of the MIDlet in the exact same manner that it would for other Media Objects following steps 1 to 6 (inclusive) as specified within this document, this includes the retrieval of any associated licenses as specified using the `license` element. However at step 3 "Capabilities Check" the following checks MUST be performed:
  - The Download Agent MUST perform its own capability checks as defined in section 5.2.2.
  - If after performing these checks the Download Agent determines it has the capabilities to download and install the MIDlet the Download Agent MUST pass the unencoded contents of the environment element (i.e. the JAD) to the MIDP AMS.
  - If the Download Agent determines it does not have the capability to download and install the MIDlet it MUST NOT pass the JAD to the MIDP AMS and it MUST not attempt to download the MIDlet. If the `installNotifyURI` element is present in the Download Descriptor the Download Agent MUST post the appropriate status report.
  - Upon receiving the JAD from the Download Agent the MIDP AMS MUST only check if it has the capabilities to install the MIDlet, that is, the MIDP AMS MUST NOT attempt to download and install the MIDlet.
  - After examining the JAD the MIDP AMS MUST indicate to the Download Agent if it has the capabilities to install the MIDlet. The MIDP AMS MUST use the format and values defined in [MIDP20] for status reports to indicate if it can install the MIDlet, i.e. if the MIDP AMS can install the MIDlet it will return a 900 "OK" status report.
  - If the MIDP AMS indicates that it can install the MIDlet then the Download Agent MUST proceed as normal to step 4.
  - If the MIDP AMS indicates that it can not install the MIDlet (i.e. the MIDP AMS returns a status report not equal to 900 "OK") the Download Agent MUST terminate the download session. If the `installNotifyURI` is present in Download Descriptor the Download Agent MUST post a 963 "Environment Internal Status" status report with the status report returned from the MIDP AMS appended to the full status report as specified in 6.2.

  It should be noted that if the MIDlet is protected using OMA DRM version 1.0 or version 2.0 then a valid license (Rights Object) must be present on the device in order to allow the MIDP AMS to install the MIDlet. In order to reduce the risk of fraud the service provider may wish to make use of two `License` elements for OMA DRM v2 protected MIDlets, here the first `license` element should only reference limited rights (e.g. execute once), these rights will be retrieved prior to or in parallel to the download of the MIDlet, the second `license` element (with the `order` attribute equal to 'post') should reference the full or remaining rights and these will only be retrieved after successful download and installation of the MIDlet.

  To allow support for MIDlets protected using OMA DRM version 1.0 the Download Agent MUST support the X-Oma-Drm-Separate-Delivery header when retrieving the Media Object as specified in [OMADRMv1]

- Upon successfully reaching Step 7 the Download Agent MUST provide the unencoded contents of the environment element (i.e. the JAD) and the downloaded JAR (or DCF containing the JAR) to the MIDP Application Management Software as (AMS) as defined in [MIDP20].
- Upon receiving the JAD and JAR the MIDP AMS should install the MIDlet according to [MIDP20] and report the appropriate status to the Download Agent using the format and values defined in [MIDP20] for status reports. If the MIDlet-Install-Notify, MIDlet-Delete-Notify or Oma-Drm-Delivery-Notify attribute is present in the JAD the MIDP AMS MUST ignore these attributes and MUST NOT post either an installation notification, a delete notification or a download complete notification.

- After receiving the status of the installation of the MIDlet from the MIDP AMS the Download Agent MUST perform step 8; Sending Installation Notification. The type of status report to post to the installNotifyURI is dependant on the success or failure of the installation of the MIDlet, therefore:
    - If the `installNotifyURI` is present in the DD and the Download Agent receives a 900 "OK" status report from the MIDP AMS then the Download Agent MUST post a 900 OK status report to the installNotifyURI.
    - If the `installNotifyURI` is present in DD and the Download Agent receives a status report from the MIDP AMS that is not equal to 900 "OK" then the Download Agent MUST post a 963 "Environment Internal Status" status report with the status report returned from the MIDP AMS appended to the full status report as specified in 6.2.
- If the MIDP AMS indicated that it successfully installed i.e. the Download Agent received a 900 "OK" status report from the MIDP AMS then the Download Agent MUST continue with the download process following the procedures in steps 9 and 10. If the MIDP AMS indicated that the installation was unsuccessful i.e. the status report was not equal to 900 "OK" then the Download Agent should continue as specified in step 9 only and provide an appropriate indication to the user.

Note: The `environment` element can only occur once per Media Object and therefore using the `environment` element to wrap a MIDP JAD means that it is not possible to use the `environment` element for any other purpose.

When the `environment` element is used to wrap a JAD the Download Agent the MIDP AMS MUST NOT display any dialogs or request any user input that would otherwise be duplicated by the Download Agent.

11.Backwards Compatibility

Devices implementing DLOTAv2 MUST support DLOTAv1 [DLOTAv1] in order to provide backwards compatibility.

Servers SHOULD respond with highest version of DD file available if client publishes support for it.

Example:

If client publishes support of DD 1.0 and DD 2.0, server has DD 1.0 and DD 2.0 then server SHOULD respond with DD 2.0.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-Download-OTA-V1_0-20040625-A	25 Jun 2004	Initial document to address the basic starting point
OMA-TS-DLOTA-V2_0-20110329-A	29 Mar 2011	Status changed to Approved by TP: OMA-TP-2011-0100-INP_Download_OTA_V2_0_ERP_for_Final_Approval
OMA-TS-DLOTA-V2_0_1-20110502-A	02 May 2011	Notice sent to TP of minor update TP ref # OMA-TP-2011-0155-INP_DLOTA_V2_0_1_ERP_for_notification

Appendix B. Static Conformance Requirements (Normative)

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

B.1 Client Conformance Requirements

The table below enumerates the client conformance requirements on OMA Download OTA version 2.0. The Enabler Release Definition for OMA Download OTA version 2.0 defines the mandatory features to be supported by the OMA Download OTA version 2.0 clients.

Item	Function	Reference	Status	Requirement
DLOTA-C-001	Support for separate delivery of Download Descriptor and Media Object	5	M	
DLOTA-C-002	Support for multiple Media Objects grouped as a Product	5	O	
DLOTA-C-003	Support for multiple Media Objects grouped as a Compound Product	5	O	
DLOTA-C-004	Support for multiple Products per one Download Descriptor	5	O	
DLOTA-C-005	Support for MMS to fetch Download Descriptors	5.1	O	MMS:MCF
DLOTA-C-006	Support for HTTP	5.1.1	M	
DLOTA-C-007	Support for TLS	5.1.1	M	
DLOTA-C-008	Support for WSP	5.1.1	O	WSP:MCF AND DLOTA-C-009
DLOTA-C-009	Support for WTLS	5.1.1	O	WTLS:MCF
DLOTA-C-010	Use of Content-Type parameter to notify the Download Descriptor media type	5.1.1	M	
DLOTA-C-011	Support for multiple Download Descriptors into one and the same transport entity	5.1.1	O	
DLOTA-C-012	Co-Delivery of Download Descriptor and Media Objects	5.1.1.1	O	
DLOTA-C-013	Support DDVersion.	5.2.1	M	
DLOTA-C-014	Download Descriptor processing rules	5.2.1.1	M	
DLOTA-C-015	Capability checking	5.2.2	M	
DLOTA-C-016	Support for user confirmation	5.2.3	M	
DLOTA-C-017	Media Object retrieval	5.2.4	M	
DLOTA-C-018	Support for Media Object retrieval of download timing reservation	5.2.4.1	O	
DLOTA-C-019	Support for Media Object update using objectID/objectVersion	5.2.4.2	O	

Item	Function	Reference	Status	Requirement
DLOTA-C-020	Support for Media Object update using Etag	5.2.4.2	O	
DLOTA-C-021	Support for pause and resume Media Object retrieval	5.2.4.3	O	
DLOTA-C-022	Support for chunked Media Object retrieval	5.2.4.4.	O	
DLOTA-C-023	Media Object retrieval from multiple servers	5.2.4.5	O	
DLOTA-C-024	License Retrieval	5.2.5	O	
DLOTA-C-025	Support for OMA DRMv2	5.2.5	O	DLOTA-C-024
DLOTA-C-026	Support for status report mechanism	5.3	M	
DLOTA-C-027	Media Object Installation	5.3.1	M	
DLOTA-C-028	Downloading of Post Licenses	5.3.2	O	
DLOTA-C-029	Media Object installation parameter	5.3.1.1	O	
DLOTA-C-030	Prevent access to Media Object until status report succeeds (at least one well-intentioned attempt) unless the content contains the `progressiveDownload Flag` with the value set to 'true'	5.3.2	M	
DLOTA-C-031	Download Confirmation	5.4.1	O	
DLOTA-C-032	Navigate browser to nextURL	5.4.1	M	WAESpec:MCF
DLOTA-C-033	Navigate browser to support URI	5.4.1	O	WAESpec:MCF
DLOTA-C-034	Support for Deletion Notification	5.4.2	O	
DLOTA-C-035	Persistent storage of Download Descriptor Elements	5.4.4	M	
DLOTA-C-036	Advertise the support of the media type of Download Descriptor.	5.5.1	M	
DLOTA-C-037	Advertise the client capability using UAProf	5.5.1.2	O	UAProf:MCF
DLOTA-C-038	Support for server initiated automatic download	5.6	O	DLOTA-C-039
DLOTA-C-039	Support for WAP Push	5.6	O	WAP Push:MCF
DLOTA-C-040	Support for reservation related notification	6.1	M	
DLOTA-C-041	Support for download completion related notification	5.2.6, 6.1	O	
DLOTA-C-042	Support for installation related notification	6.1	M	

Item	Function	Reference	Status	Requirement
DLOTA-C-043	Support for deletion related notifications	6.1	O	
DLOTA-C-044	Proper formatting of status report	6.2	M	
DLOTA-C-045	Support for Download Descriptor	7	M	
DLOTA-C-046	Proper formatting of Download Descriptor	7.2	M	
DLOTA-C-047	Support for the environment element	7.2.3.2.10	O	
DLOTA-C-048	Support for progressive download	7.2.3.3.7	O	
DLOTA-C-049	Support for validity time	7.2.3.3.11	O	
DLOTA-C-050	Download Descriptor Extensions	7.3	M	
DLOTA-C-051	Support for HTTP digest authentication	8.1	M	
DLOTA-C-052	Support for GAA	8.1	O	
DLOTA-C-053	Support for Server Authentication	8.2	M	
DLOTA-C-054	Support for white lists	8.3	M	
DLOTA-C-055	Support for Device Management to provision white lists.	8.3	O	OMADM:MCF
DLOTA-C-056	Support for confidentiality protection	8.4	M	
DLOTA-C-057	Support for WAP TLS Profile	8.4	M	
DLOTA-C-058	Mandatory cipher suites	8.4	M	
DLOTA-C-059	Additional cipher suites	8.4	O	
DLOTA-C-060	Support for Java MIDP OTA	10	O	DLOTA-C-047
DLOTA-C-061	Support for DLOTAv1 for backwards compatibility	11	M	DLOTAv1:MCF

B.2 Server Conformance Requirements

The table below enumerates the server conformance requirements on OMA Download OTA version 2.0. The Enabler Release Definition for OMA Download OTA version 2.0 defines the mandatory features to be supported by the OMA Download OTA version 2.0 servers.

Item	Function	Reference	Status	Requirement
DLOTA-S-001	Support for separate delivery of Download Descriptor and Media Object	5	M	
DLOTA-S-002	Support for multiple Media Objects grouped as a Product	5	M	
DLOTA-S-003	Support for multiple Media Objects grouped as a Compound Product	5	M	

Item	Function	Reference	Status	Requirement
DLOTA-S-004	Support for multiple Products per one Download Descriptor	5	M	
DLOTA-S-005	Support for HTTP	5.1.1	M	
DLOTA-S-006	Support for TLS	5.1.1	M	
DLOTA-S-007	Support for WSP	5.1.1	O	WSP:MSF AND DLOTA-S-008
DLOTA-S-008	Support for WTLS	5.1.1	O	WTLS:MSF
DLOTA-S-009	Use of Content-Type parameter to notify the Download Descriptor media type	5.1.1	M	
DLOTA-S-010	Support for multiple Download Descriptors into one and the same transport entity	5.1.1	O	
DLOTA-S-011	Co-Delivery of Download Descriptor and Media Objects	5.1.1.1	O	
DLOTA-S-012	Media Object retrieval	5.2.4	M	
DLOTA-S-013	Support for Media Object retrieval of download timing reservation	5.2.4.1	O	
DLOTA-S-014	Support for Media Object update using objectID/objectVersion	5.2.4.2	O	
DLOTA-S-015	Support for Media Object update using Etag	5.2.4.2	O	
DLOTA-S-016	Support for pause and resume Media Object retrieval	5.2.4.3	O	
DLOTA-S-017	Support for chunked Media Object retrieval	5.2.4.4.	O	
DLOTA-S-018	License Retrieval	5.2.5	O	
DLOTA-S-019	Support for OMA DRMv2	5.2.5	O	DLOTA-S-018
DLOTA-S-020	Support for status report mechanism	5.3	M	
DLOTA-S-021	Support for Deletion Notification	5.4.2	O	
DLOTA-S-022	Support for content type application/vnd.oma.dd2+xml.	5.5.1.1	M	
DLOTA-S-023	Support for server initiated automatic download	5.6	O	DLOTA-S-024
DLOTA-S-024	Support for WAP Push	5.6	O	WAP Push:MSF
DLOTA-S-025	Support for reservation related notification	6.1	O	
DLOTA-S-026	Support for download completion related notification	5.2.6, 6.1	O	
DLOTA-S-027	Support for installation related notification	6.1	M	

Item	Function	Reference	Status	Requirement
DLOTA-S-028	Support for deletion related notifications	6.1	O	
DLOTA-S-029	Support for Download Descriptor	7	M	
DLOTA-S-030	Proper formatting of Download Descriptor	7.2	M	
DLOTA-S-031	Support for the environment element	7.2.3.2.10	O	
DLOTA-S-032	Support for HTTP digest authentication	8.1	M	
DLOTA-S-033	Support for GAA	8.1	O	
DLOTA-S-034	Support for Server Authentication	8.2	M	
DLOTA-S-035	Support for Device Management to provision white lists.	8.3	O	OMADM:MCF
DLOTA-S-036	Support for confidentiality protection	8	M	
DLOTA-S-037	Support for WAP TLS Profile	8.4	M	
DLOTA-S-038	Mandatory cipher suites	8.4	M	
DLOTA-C-039	Support for Java MIDP OTA	10	O	DLOTA-S-031
DLOTA-S-040	Respond with highest version of DD file.	11	O	

Appendix C. Difference between Compound Product and Product

Table 5,
Figure 3 and
Figure 4 highlight the difference between Compound Product and Product.

A Compound Product is Product where all Licenses and Media Objects must be Downloaded and installed as one entity (Figure3). If one of the License or one of the Media Objects fail to be downloaded or installed, the complete Compound Product SHALL be discarded by the Download Agent. Once the Compound Product is installed, each Media Object in a Product is handled as an independent Media Object.

A Product is a logical grouping of one or more independent Media Objects and optional associated Licenses that are downloaded and installed as one logical group (
Figure 4). This can be an arbitrary set defined by a content provider. Once the product is installed, each Media Object in a Product is handled as an independent Media Object.

	Compound Product	Product
Capability Check	Capability Check must be performed as if it is a single Media Object.	Capability Check must be performed separately for each Media Object
Installation	All Media Objects must be installed as single entity.	Each Media Object can be install separately.
Download Completion Notification	The Download Completion Notification must be specified for the Product level only.	The Download Completion Notification can be specified for separate Media Objects.
Installation Notification	The Installation Notification must be specified for the Product level only.	The Installation Notification can be specified for separate Media Objects.
Deletion Notification	The Deletion Notification can be specified for separate Media Objects.	

Table 5: Difference between Compound Product and Product

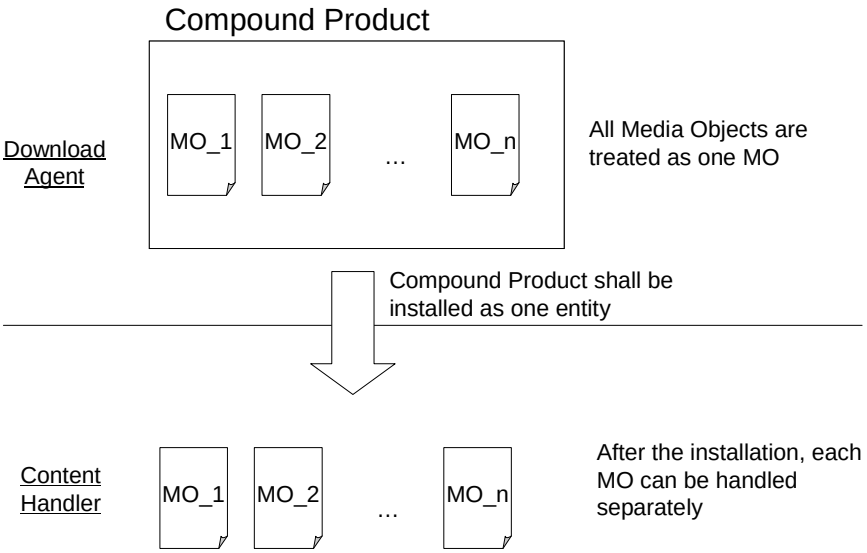

Figure 3: Compound Product

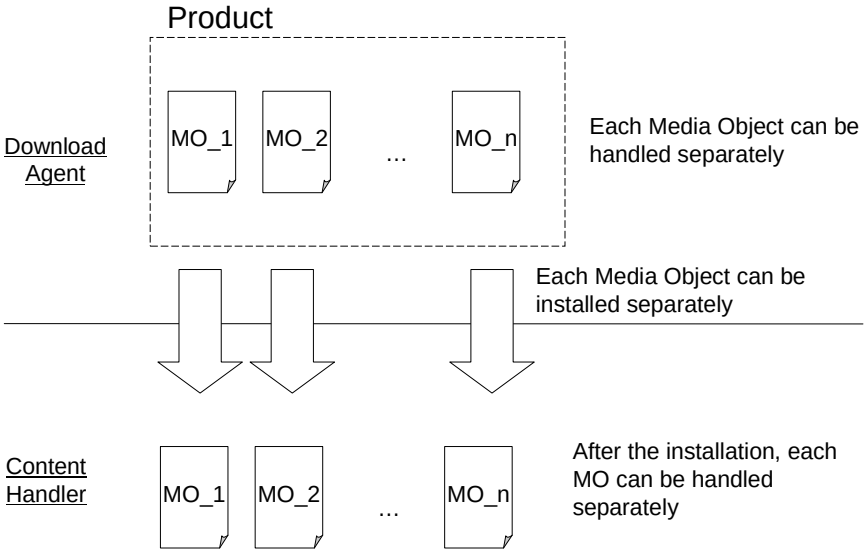

Figure 4: Product

Appendix D. Example of Download Descriptor (Informative)

D.1 Example of Basic Download Descriptor

```
<media xmlns="urn:oma:xml:dl:dd:2.0" DDVersion=”2.0”>
  <product>
    <mediaObject>
      <size>100</size>
      <type>image/gif</type>
      <objectID>cid:image@example.com</objectID>
      <objectURI>
         <server>http://download.example.com/image.gif</server>
      </objectURI>
    </mediaObject>
  </product>
</media>
```

D.2 Example of Basic Download Descriptor with installNotifyURI

```
<media xmlns="urn:oma:xml:dl:dd:2.0" DDVersion=”2.0”>
  <product>
    <mediaObject>
      <meta>
        <installNotifyURI>http://download.example.com/
              image.gif?id=image</installNotifyURI>
      </meta>
      <size>100</size>
      <type>image/gif</type>
      <objectID>cid:image@example.com</objectID>
      <objectURI>
         <server>http://download.example.com/image.gif</server>
      </objectURI>
    </mediaObject>
  </product>
</media>
```

D.3 Example for multiple server support

```
<media xmlns="urn:oma:xml:dl:dd:2.0" DDVersion=”2.0”>
  <product>
    <mediaObject>
      <meta>
        <installNotifyURI>http://download.example.com/
              image.gif?id=image</installNotifyURI>
      </meta>
      <size>100</size>
      <type>image/gif</type>
      <objectID>cid:image@example.com</objectID>
      <objectURI>
         <server>http://download.example.com/image.gif</server>
         <server>http://download.example.alt1.com/image.gif</server>
         <server>http://download.example.alt2.com/image.gif</server>
      </objectURI>
    </mediaObject>
  </product>
</media>
```

D.4 Example of environment element

Company “acme” developed a game on the software platform specified by “whatever” standardization forum. This standardization forum defines a XML schema of internal structure for the DLOTA DD `environment` element together with the identifier of the content handler (*envtype* value) that knows the platform specific internal format of the package. The XML schema is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
      targetNamespace=http://www.whatever.org/xmlns/dd/vx.y
      xmlns:md=http://www.whatever.org/xmlns/dd/vx.y
```

```
      xmlns:xsd=http://www.w3.org/2001/XMLSchema
      elementFormDefault="qualified">
      <xsd:element name="option1" type="xsd:string"/>
      <xsd:element name="option2" type="xsd:string"/>
</xsd:schema>
```

The DLOTA Download Descriptor of the game application is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       xmlns:dd="http://www.whatever.org/xmlns/dd/vx.y"
       DDVersion=”2.0”>
  <vendor>
    <name>acme</name>
  </vendor>
  <product>
    <mediaObject>
      <meta>
        <name>game.exe</name>
        <environment
          envtype="http://www.whatever.org/xmlns/dd/ResourcePackage" >
          <dd:option1>v6.4</dd:option1>
          <dd:option2>ABCDEF</dd:option2>
        </environment>
      </meta>
      <size>1234</size>
      <type>application/executable</type>
      <objectID>cid:game@musicvendor.com</objectID>
      <objectURI>
         <server>http://acme.com/game.exe</server>
      </objectURI>
    </mediaObject>
  </product>
</media>
```

D.5 Example of environment element use for wrapping a MIDP JAD

Given the MIDP 2.0 schema defined section 10 and the following JAD:

MIDlet-1:MIDPgame,/icon.png, MIDPgameMIDlet
MIDlet-Description: A MIDP2.0 Game
MIDlet-Icon:/icon.png
MIDlet-Jar-Size:13511
MIDlet-Jar-URL:http://www.someMIDPVendor.com/MIDPgame.jar
MIDlet-Name: MIPDgame
MIDlet-Vendor:SomeMIDPVendor
MIDlet-Version:0.0.2
MicroEdition-Configuration:CLDC-1.0
MicroEdition-Profile:MIDP-2.0
The corresponding DD will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       xmlns:dd="urn:oma:xml:dl:jad:2.0"
       DDVersion=”2.0”>
  <vendor>
    <name>SomeMIDPVendor</name>
  </vendor>
  <product>
    <mediaObject>
      <meta>
        <name>MIDPgame.jar</name>
        <description>A MIDP2.0 Game</description>
```

```
       <installNotifyURI>http://www.someMIDPVendor.com/status.asp?ID=1234
       </installNotifyURI>
       <environment
          envtype="urn:oma:xml:dl:jad:2.0" >
          <dd:jad>
TUlEbGV0LTE6TUlEUGdhbWUsL2ljb24ucG5nLCBNSURQZ2FtZU1JRGxldA0KTUlEbGV0LURlc2NyaXB0aW9uOiBBIE
1JRFAyLjAgR2FtZQ0KTUlEbGV0LUljb246L2ljb24ucG5nDQpNSURsZXQtSmFyLVNpemU6MTM1MTENCk1JRGxldC1K
YXItVVJMOmh0dHA6Ly93d3cuc29tZU1JRFBWZW5kb3IuY29tL01JRFBnYW1lLmphciANCk1JRGxldC1OYW1lOiBNSV
BEZ2FtZQ0KTUlEbGV0LVZlbmRvcjpTb21lTUlEUFZlbmRvcg0KTUlEbGV0LVZlcnNpb246MC4wLjINCk1pY3JvRWRp
dGlvbi1Db25maWd1cmF0aW9uOkNMREMtMS4wDQpNaWNyb0VkaXRpb24tUHJvZmlsZTpNSURQLTIuMA==
          </dd:jad>
       </environment>
      </meta>
      <size>13511</size>
      <type>application/java-archive</type>
      <objectURI>http://www.someMIDPVendor.com/MIDPgame.jar</objectURI>
    </mediaObject>
  </product>
</media>
```

D.6 Example with multiple Products and Media Objects

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       ddVersion="2.0">
  <product>
    <meta>
      <name>Product1</name>
      <description>Cheap Product</description>
    </meta>
    <mediaObject>
      <meta>
        <name>Being anonymous</name>
      </meta>
      <size>6034500</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234567@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234567.3g2</server>
      </objectURI>
    </mediaObject>
    <mediaObject>
      <meta>
        <name>Nobody knows me</name>
      </meta>
      <size>60236476</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234568@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234568.3g2</server>
      </objectURI>
    </mediaObject>
  </product>
  <product>
    <meta>
      <name>Product2</name>
      <description>Expensive Product</description>
    </meta>
    <mediaObject>
      <meta>
        <name>Somebody thinks he knows me</name>
      </meta>
      <size>6034500</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234569@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234569.3g2</server>
      </objectURI>
    </mediaObject>
  </product>
  <nextURL>http://www.musicvendor.com/shop?nextPage</nextURL>
</media>
```

D.7 Example with license elements

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
       xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
       ddVersion="2.0">
  <product>
    <meta>
      <name>Product1</name>
      <description>Cheap Product</description>
      <license lictype="x-wap-application:drm.ua">
        <roap:roapTrigger xsi:type="roap:RoapTrigger">
         <roAcquisition>
            <riID>
              <keyIdentifier xsi:type="roap:X509SPKIHash">
                <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
              </keyIdentifier>
            </riID>
            <roapURL>http://ri.example.com/ro.cgi?tid=qw683hgew7d</roapURL>
            <roID>roId</roID>
            <contentID>cid:product1@musicvendor.com</contentID>
          </roAcquisition>
        </roap:roapTrigger>
      </license>
    </meta>
    <mediaObject>
      <meta>
        <name>Being anonymous</name>
        <license lictype="x-wap-application:drm.ua">
          <roap:roapTrigger xsi:type="roap:RoapTrigger">
            <roAcquisition>
              <riID>
                <keyIdentifier xsi:type="roap:X509SPKIHash">
                  <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
                </keyIdentifier>
              </riID>
              <roapURL>http://ri.example.com/ro.cgi?tid=qw683hgew7d</roapURL>
              <roID>roId</roID>
              <contentID>cid:7654321@musicvendor.com</contentID>
            </roAcquisition>
          </roap:roapTrigger>
        </license>
      </meta>
      <size>6034500</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234567@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234567.3g2</server>
      </objectURI>
    </mediaObject>
    <mediaObject>
      <meta>
        <name>Nobody knows me</name>
      </meta>
      <size>60236476</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234568@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234568.3g2</server>
      </objectURI>
    </mediaObject>
  </product>
  <nextURL>http://www.musicvendor.com/shop?nextPage</nextURL>
</media>
```

D.8 Example with additional textual metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       ddVersion="2.0">
  <vendor>
```

```
    <name>MusicVendor</name>
    <home>http://www.musicvendor.com</home>
    <logo>http://www.musicvendor.com/logo.jpg</logo>
    <support>http://www.musicvendor.com/support.html?dd12345</support>
  </vendor>
  <product>
    <meta>
      <name>Product1</name>
      <description>Cheap Product</description>
      <text id="artist" display="Artist">John Doe</text>
      <text id="album" display="Album">Songs for the Unknown</text>
      <infoURL>http://www.JohnDoe.Com</infoURL>
      <iconURI>http://www.JohnDoe.Com/coverArt.jpg</iconURI>
    </meta>
    <mediaObject>
      <meta>
        <name>Being anonymous</name>
        <!--  artist and album are inherited from Product meta -->
      </meta>
      <size>6034500</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234567@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234567.3g2</server>
      </objectURI>
    </mediaObject>
    <mediaObject>
      <meta>
        <name>Nobody knows me</name>
        <!--  artist and album are inherited from Product meta -->
      </meta>
      <size>60236476</size>
      <type>audio/3gpp</type>
      <objectID>cid:1234568@musicvendor.com</objectID>
      <objectURI>
         <server>http://www.musicvendor.com/1234568.3g2</server>
      </objectURI>
    </mediaObject>
  </product>
  <nextURL>http://www.musicvendor.com/shop?nextPage</nextURL>
</media>
```

D.9 Example with license element for acquiring Rights only (Superdistribution).

```
<?xml version="1.0" encoding="UTF-8"?>
<media xmlns="urn:oma:xml:dl:dd:2.0"
       xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
       xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
       ddVersion="2.0">
  <product>
    <meta>
      <name>Product1</name>
      <description>Cheap Product</description>
      <installNotifyURI>http://ri.example.com/roAck.cgi?tid=qw683hgew7d</installNotifyURI>
      <license lictype="x-wap-application:drm.ua">
        <roap:roapTrigger xsi:type="roap:RoapTrigger">
          <roAcquisition>
            <riID>
              <keyIdentifier xsi:type="roap:X509SPKIHash">
                <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
              </keyIdentifier>
            </riID>
            <roapURL>http://ri.example.com/ro.cgi?tid=qw683hgew7d</roapURL>
            <roID>roId</roID>
            <contentID>cid:product1@musicvendor.com</contentID>
          </roAcquisition>
        </roap:roapTrigger>
      </license>
      <suppressUserConfirmation>Always</suppressUserConfirmation>
    </meta>
```

```
  </product>
  <nextURL>http://www.musicvendor.com/shop?nextPage</nextURL>
</media>
```

Appendix E. Example of Download Transaction (Informative)

The example below shows a very simple use case of a download transaction.

E.1 HTTP Request to view a download service page

When requesting the rendering a service page, the request might look as follows:

```
GET http://www.service.com/download_service.html
Host: www.service.com
Accept: image/gif, multipart/mixed, application/vnd.oma.dd2+xml, text/html
```

The response from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Content-Length: 2543
Content-Type: text/html

<?xml version="1.0"?>
<!DOCTYPE      html      PUBLIC      "-//WAPFORUM//DTD      XHTML      Mobile      1.0//EN"
"http://www.openmobilealliance.org/tech/DTD/xhtml-mobile10.dtd" >
      <html xmlns="http://www.w3.org/1999/xhtml" >
      <head>
           <title>Service presentation</title>
           <base href="http://host.foo.bar/" />
      </head>
      <body>
           <p>Please select the object <a href="pic-dir/picture.dd?ID=1234">
             here</a>!</p>
      </body>
</html>
```

E.2 HTTP Request for Download Descriptor

When requesting the download of a download descriptor, the request headers might look as follows:

```
GET http://host.foo.bar/pic-dir/picture.dd?ID=1234
Host: host.foo.bar
Accept: application/vnd.oma.dd2+xml
User-Agent: CoolPhone/1.4
Accept-Language: en-US, fi, fr
Accept-Charset: utf-8
```

The response from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Content-Length: 393
Content-Type: application/vnd.oma.dd2+xml; charset=utf-8

<media xmlns="urn:oma:xml:dl:dd:2.0" DDVersion=”2.0”>
  <product>
    <mediaObject>
      <meta>
        <installNotifyURI>http://download.example.com/
              image.gif?id=image</installNotifyURI>
      </meta>
      <size>100</size>
      <type>image/gif</type>
      <objectID>cid:image@example.com</objectID>
      <objectURI>
         <server>http://download.example.com/image.gif</server>
      </objectURI>
    </mediaObject>
  </product>
</media>
```

E.3 HTTP Request to Install a Media Object

When requesting the download of a Media Object file, the request might look as follows:

```
GET http://host.foo.bar/pic-dir/picture.gif
Host: host.foo.bar
Accept: image/gif, image/jpg
```

The response from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Content-Length: 25432
Content-Type: image/gif

… GIF picture…
```

E.4 Install Status via HTTP Post Request

After a successful reception of the GIF, the following would be posted:

```
POST http://foo.bar.com/status
Host: foo.bar.com
Content-Length: 13

900 Success
```

Example Status report for a Product containing several Media Objects with License:

```
POST http://foo.bar.com/status
Host: foo.bar.com
Content-Length: 196

970 Mixed Status
956 License Retrieval Succeeded
cid:id1234@company.com 900 Success
cid:id1234@company.com 956 License Retrieval Succeess
cid:id5678@company.com 953 Non-Acceptable Content
```

The response from the server might be:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
```

E.5 Pause and Resume Media Retrieval

When requesting the download of a Media Object file, the request might look as follows:

```
GET http://host.foo.bar/pic-dir/picture.gif
Host: host.foo.bar
Accept: image/gif, image/jpg
```

The response from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Content-Length: 25432
Content-Type: image/gif
Accept-Ranges: bytes
ETag: fasd987sadf98@example.com/1.1

… GIF picture…
```

1034 bytes has been downloaded but the connection is lost for some reason.

The Media Object Retrieval is resumed

```
GET http://host.foo.bar/pic-dir/picture.gif
Host: host.foo.bar
Accept: image/gif, image/jpg
Range: 1034-25431
If-Match: fasd987sadf98@example.com/1.1
```

The response from server might look as follows:

```
HTTP/1.1 206 Partial Content
Server: CoolServer/1.3.12
Content-range: bytes 1034-25431/25432
Content-Type: image/gif
Accept-Ranges: bytes
ETag: fasd987sadf98@example.com/1.1

… GIF picture…
```

Appendix F. Media Type Registration

This appendix describes the registration of MIME media type application/vnd.oma.dd2+xml

The MIME media type for a Download Descriptor v2 is registered in IANA (Internet Assigned Number Authority). You can also see the registered reference at http://www.iana.org/assignments/media-types/application/vnd.oma.dd2+xml.

1	**MIME Media Type Name :**	Application
2	**MIME subtype name :**	vnd.oma.dd2+xml
3	**Required parameters :**	none
4	**Optional parameters :**	charset : This parameter has identical semantics to the charset parameter specified in [XMLMIME]. version Indicates the Download Descriptor version. The value has the format: <major>.<minor>; where major and minor are integers. For example, version="2.0" indicates version 2.0.
5	**Encoding considerations :**	binary This media type may require encoding on transports not capable of handling binary.
6	**Security considerations :**	The vnd.oma.dd2+xml typed content (i.e. Download Descriptor v2.0) contains meta data for the media object that is to be downloaded by the Download Agent. It also can optionally contain executable content in the form of installation commands. In normal usage the vnd.oma.dd2+xml typed content does not contain information that needs to be kept private. The vnd.oma.dd2+xml typed content itself does not provide either privacy or integrity protection. However, the specification [OMADLOTAv2] provides Download Agent authentication and Server authentication using HTTP Digest mechanism [RFC2617] and TLS [RFC2246], and also provides confidentiality and integrity protection using TLS [RFC2246]. For more details, see [XMLMIME] and [OMADLOTAv2].
7	**Interoperability considerations :**	The OMA Download specifications [OMADLOTAv2] specify user agent (Download Agent) conformance rules that dictate behaviour that must be followed when dealing with, among other things, unrecognized elements.
8	**Published specification :**	The OMA Download OTA v2 specification is published at

<table>
<tr><td></td><td colspan="2"></td><td><http://www.openmobilealliance.org/>.</td></tr>
<tr><td>9</td><td colspan="2">Applications which use this media type :</td><td>OMA Download Agents. See [OMADLOTAv2].</td></tr>
<tr><td>10</td><td colspan="2">Additional information :</td><td></td></tr>
<tr><td></td><td>Magic number(s) :</td><td></td><td>There is no single initial byte sequence.</td></tr>
<tr><td></td><td>File extension(s) :</td><td></td><td>.xml, .dd2</td></tr>
<tr><td></td><td>Macintosh File Type Code(s) :</td><td></td><td>TEXT</td></tr>
<tr><td></td><td>Object Identifier(s) or OID(s) :</td><td></td><td>none</td></tr>
<tr><td>11</td><td colspan="2">Intended usage :</td><td>Common</td></tr>
<tr><td>12</td><td colspan="2">Other Information/General Comment :</td><td>For the encoding considerations, see [XMLMIME] and [OMADLOTAv2].

References:<table><tr><td>[OMADLOTAv2]</td><td>“OMA Download OTA V2.0 Specification”, Open Mobile Alliance Specification. Available at <http://www.openmobilealliance.org/>.</td></tr><tr><td>[XMLMIME]</td><td>Murata, M., St.Laurent, S., Kohn, D., "XML Media Types", RFC3023, January 2001.</td></tr></table></td></tr>
<tr><td>13</td><td colspan="2">Person to contact for further information :</td><td>The OMA Download specifications are a work product of the Open Mobile Alliance's BAC DLDRM Working Group. The Open Mobile Alliance has change control over these specifications. mailto:technical-comments@mail.openmobilealliance.org</td></tr>
</table>