Generic Content Download Over The Air Specification

Version 1.0

Version 21-Feb-2003

Open Mobile Alliance
OMA-Download-OTA-v1_0-20030221-C

A list of errata and updates to this document is available from the Open Mobile Alliance™ Web site, http://www.openmobilealliance.org/, in the form of SIN documents, which are subject to revision or removal without notice.

.

A list of errata and updates to this document is available from the OMA™ Web site, http://www.openmobilealliance.org/, in the form of SIN documents, which are subject to revision or removal without notice.

This document is available online in PDF format at http://www.openmobilealliance.org/.

Known problems associated with this document are published at http://www.openmobilealliance.org/.

Comments regarding this document can be submitted to the Open Mobile Alliance™ in the manner published at http://www.openmobilealliance.org/technical.htm

Document History	
OMA-Download-OTA-v1_0-20030221-C	Current
OMA-Download-OTA-v1_0-20021219-c	TP approved
OMA-Download-OTA-v1_0-20021205-c	Class 2 Changes submitted for TP approval
OMA-Download-OTA-v1_0-20020912-c	Initial Candidate
OMA-Download-OTA-v1_0-20020912-p	Proposed

Contents

1. Scope

Open Mobile Alliance (OMA) Wireless Application Protocol (WAP) is a result of continuous work to define an industry-wide specification for developing applications that operate over wireless communication networks. The scope for the Open Mobile Alliance is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation, and fast and flexible service creation, WAP defines a set of protocols in transport, session, and application layers. For additional information on the WAP architecture, refer to [WAPARCH].

This specification defines application-level protocols and behaviours needed to provide both powerful as well as reliable content download functionality. It is anticipated that this will be an enabler for billing, and thus make premium priced content available to mobile users.

The specification addresses requirements such as Content Discovery, Authentication, Delivery Negotiation, Content Delivery and Delivery Confirmation. It takes advantage of work that has been done in the above-mentioned areas in both the Open Mobile Alliance as well as in the Java Community Process (sm).

The technology defined in this specification provides a consistent download interface and functionality for all generic content types, to complement the download of Java™ applications that has been defined in [MIDPOTA].

The intended audience of this specification are implementers of Download Agents and download servers, as well as other people who have some in-depth interest in the download procedures. It is not intended to be a tutorial for application authors.

2. References

2.1. Normative References

[CREQ] "Specification of WAP Conformance Requirements". WAP Forum™. WAP-221-CREQ, http//www.openmobilealliance.org/

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

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

[RFC2046] Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, http://www.ietf.org

[RFC2387] "The MIME Multipart/Related Content-type", E. Levinson, 1998, http://www.ietf.org/

[RFC2616] "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, R. Fielding, et al., June 1999. http://www.ieft.org/rfc/rfc2616.txt

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

[W-HTTP] " Wireless Profiled HTTP", WAP-229-HTTP, WAP Forum™, http://www.openmobilealliance.org/

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

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

[XML] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, Tim Bray et al., 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006

[XMLNS] "Namespaces in XML", W3C Recommendation, Tim Bray et al., 14 January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/

[XMLSchema1] "XML Schema Part 1: Structures", W3C Recommendation, Henry S. Thompson et al., 2 May 2001. http://www.w3.org/TR/xmlschema-1/

[XMLSchema2] "XML Schema Part 2: Datatypes", W3C Recommendation, Paul V. Biron et al., 2 May 2001. http://www.w3.org/TR/xmlschema-2/

2.2. Informative References

[WAPARCH] "WAP Architecture". WAP Forum™. WAP-210-WAPARCH. http//www.openmobilealliance.org/

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

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

[RFC2617] "HTTP Authentication, basic and digest authentication mechanisms", RFC 2617, http://www.ietf.org/

[UAProf] "User Agent Profile", Open Mobile Alliance™. OMA-WAP-UAProf-v1_1. http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1. Conventions

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

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

3.2. Definitions

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.

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 Descriptor Metadata about a *media object* and instructions to the *download agent* for how to download it. The object triggers the Download Agent in the client device. It describes the media object to be downloaded and allows the client device to decide if it has the capabilities to install and render/execute the Media Object.

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 [W-HTTP] (or [WSP] if the environment is WAP 1.x). Other protocols, including full support for HTTP, may be used if supported by both parties.

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 media object 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 User Agent	A user agent in the device responsible for downloading a *media object* described by a *downloaddescriptor*. 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, and that the content (to the best knowledge of the Download Agent) will be made available to the user.
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].
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.
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.
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

CID	Content Identifier
DD	Download Descriptor
HTTP	HyperText Transfer Protocol
JAD	Java™ Application Descriptor

JAR	Java™ Archive
J2ME	Java™ 2 Micro Edition
MIDP	Mobile Information Device Profile
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
OTA	Over The Air
RP	Recommended Practices
URL	Universal Resource Locator
URI	Universal Resource Identifier
WAP	Wireless Application Protocol
XML	Extensible Markup Language

3.4. Acknowledgements

Sun, Sun Microsystems, the Sun Logo, Java, J2ME are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

4. Introduction

The mobile industry is on the verge of introducing large-scale Internet style, media type independent, download of media objects. One of the areas of deployment for this technology is e-commerce. The most likely objects to kick off the business are Ringing Tones, Screen Savers and Java™ MIDlets.

4.1. Goal

The goal of this specification is to define a technology for confirmed download that is used to deliver digital content such as entertainment and business applications as well as objects to be rendered to mobile terminals. Another important application area for download is to personalise terminals according to a user's preferences and lifestyle. Some of the specific goals include

- Enable various payment models to support launch of e-commerce concepts.
- Avoid fragmentation of content space. Content for mobile devices with different capabilities can be published using a consistent concept.
- Create commonality between the download process of all types of media: e.g. games, melodies, and pictures.
- Enable both automated as well as manual client driven capability negotiation
- Enable a mechanism that allows the initial download solution to be extended with new attributes and functionality
- Create a solution that is quick and easy to implement and deploy in order to make time to market short

4.2. Architecture

The fundamental architectural vision behind this specification is the logical separation of presentation (content) server, download server and content storage. This architecture allows for very simple presentation servers, with no special functionality to enable e-commerce. The e-commerce and download management functionality can instead be concentrated into the Download Server.

This architecture explicitly allows for both centralized deployment, where there is a strong association between presentation server and download server, as well as decentralized deployment where there is a relatively low level of integration between presentation and download servers.

The functionality enables the implementation of confirmed and reliable, and thus billable, transactions between a server entity (Presentation Server, Download Server) and a client device. The functionality allows any type of content to be downloaded.

4.3. Overview

The mechanisms for media object download defined in this specification support at least two high-level commercial use cases:

- A pay-per-transaction model where the confirmation of a successful installation of a media object typically triggers server side billing actions.
- A subscription model that does not require explicit confirmation of individual succeeded download operations. In this case of commercial service the service provider typically makes a separate agreement with an end user to download multiple media objects at the end users convenience.

4.3.1. OMA Download

The OMA Download model leverages the HTTP download mechanisms, and adds a number of features as defined in this specification. These include additional tools for content negotiation, well-formalised (protocol independent) META-data presentation and application layer confirmation of installation.

The negotiation model allows both the client device and the end user to evaluate if a download is to proceed to the phase where a data object it transferred to the device. Some of the attributes of the Download Descriptor allow the download agent to compare the currently available resources on the client device to the META-data representing the media object to be downloaded. Another set of attributes allow the end user to evaluate if the media object is worthwhile for him, if he wants to complete the transaction or abort before the media object has been transferred.

A download transaction always includes a degree of uncertainty, the client and the server cannot simultaneously know with absolute certainty that a transaction has succeeded, and that the other party is in agreement with this deduction. The logic of the OMA Download Installation Notification has been defined in such a way that the client has a slight advantage. There may be situations where the server does not know that the client has accepted the media object, but there is no situation where the server would believe that the client device has accepted the media object when it in fact has been rejected.

The generic OMA Download, based on the concept of a Download Descriptor, includes two basic scenarios:

- OMA Download with Separate Delivery of Download Descriptor and Media Object.
    - The Download Descriptor, which contains META-data related to the download transaction, is typically delivered separately from the media object it references.
- OMA Download with Co-Delivery of Download Descriptor and Media Object
    - In some cases it may be beneficial to combine the Download Descriptor and the media object in one delivery package.

In the two scenarios there may or may not be an installation notification, depending on what was requested in the Download Descriptor.

These main OMA Download scenarios are briefly described below.

OMA Download with Separate Delivery of Download Descriptor and Media Object

This download scenario is a process that includes separate request-reply interaction pairs for Download Descriptor delivery, content delivery and the optional application level transaction confirmation.

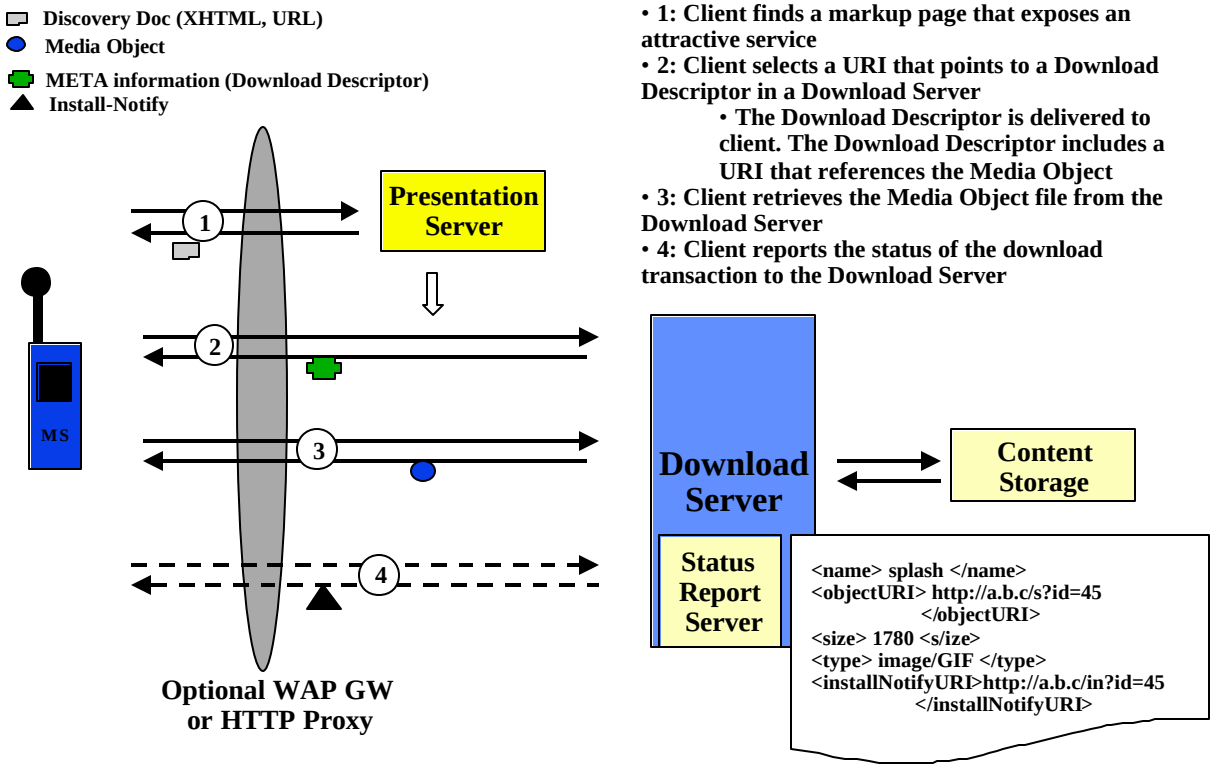

Figure 1. The full generic content download process, with separate transfer of Download Descriptor and media object. This download process can be truncated in situations where no InstallNotifyURI has been defined.

OMA Download with Co-Delivery of Download Descriptor and Media Object

This download scenario is a process where the Download Descriptor is delivered together with the Media Object within a single request-reply interaction. This delivery is then followed by optional application level transaction confirmation.

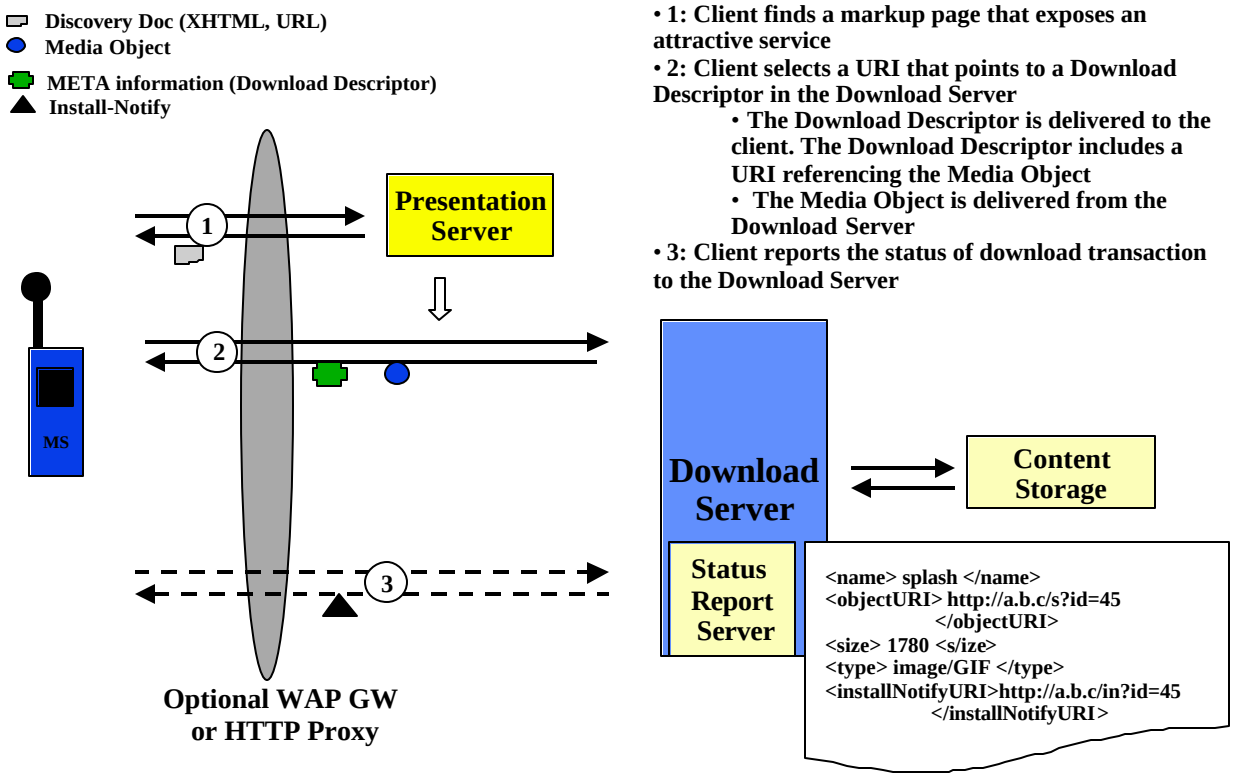

Figure 2. The shortcut of the most elaborate download process with a combined delivery of Download Descriptor and Media Object. This download process can be truncated in situations where no InstallNotifyURI has been defined.

OMA DOWNLOAD without Installation Notification

This download scenario illustrates the case when an Installation Notification is not used to provide confirmation to the server about the successful media object installation. The advantage of this scenario is that one request-reply pair can be omitted from the process if the use case does not require application level transaction confirmation.

This scenario represents an unreliable download mechanism (similar to an HTTP retrieval of content) particularly suitable for use cases where absolute reliability is less of a priority than bandwidth optimisation.

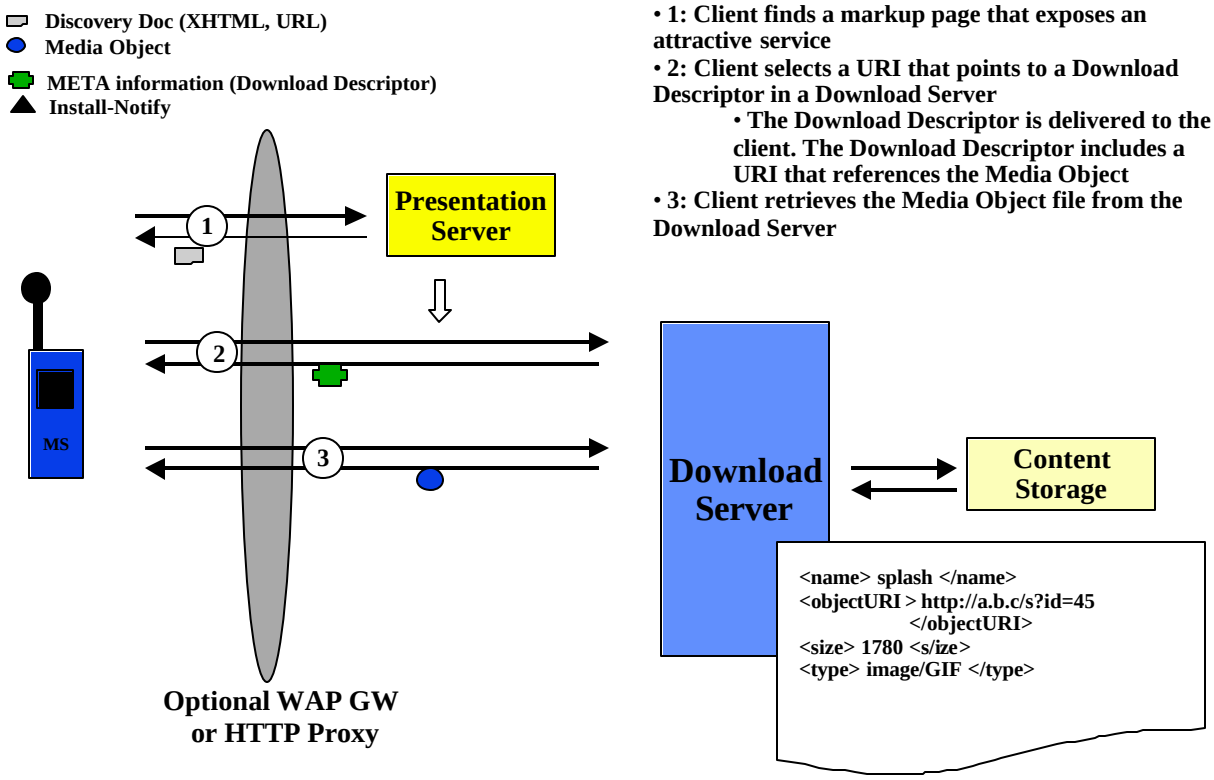

Figure 3. The download process is significantly simplified in the case that no Installation Notification is required. The over the air traffic is then reduced by one request-reply interaction.

Service flow in a browsing environment

Each of the download scenarios above can be combined with the mechanism for application flow control defined in this specification. The flow control mechanism provides the service creator with a tool that allows him to advise the client device about the next URL to access in case the end user decides to continue with a browsing interaction.

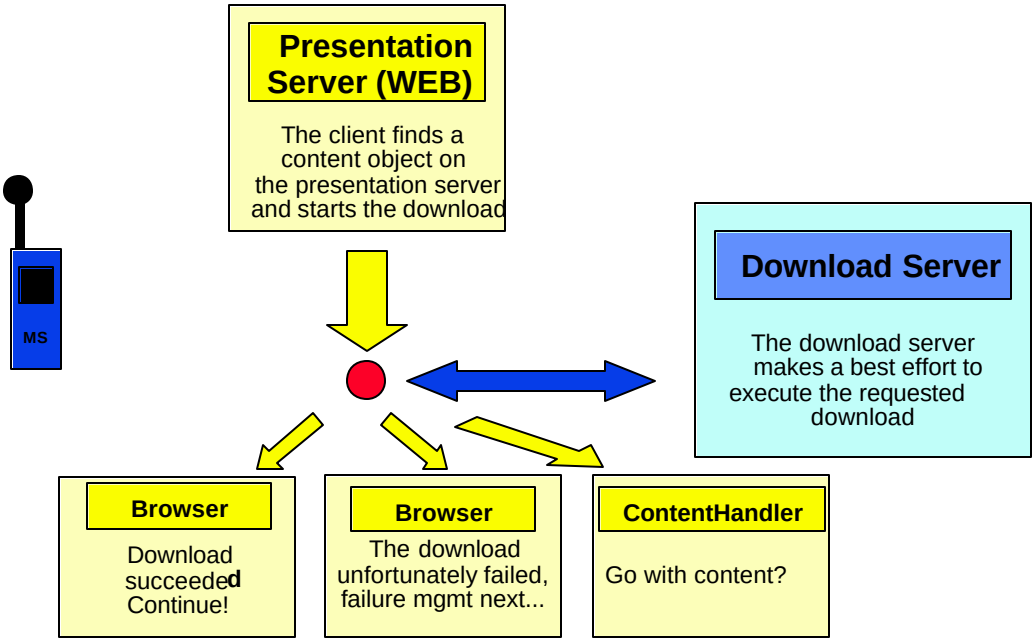

Figure 4. Basic WEB/WAP service user experience flow control.

The control flow mechanism covers the basic use case for context management where the end user selects to continue with a browsing operation after the completion of the download transaction. The mechanism can be used both in the situation where the download was completed successfully and in the situation where it was terminated with some kind of a failure.

The case when the end user after the download selects to perform some other action, like render the downloaded media object, is outside the scope of this specification.

5. OMA DOWNLOAD Process

During the download and installation process the user SHOULD be given opportunities 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 a Media Object 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.

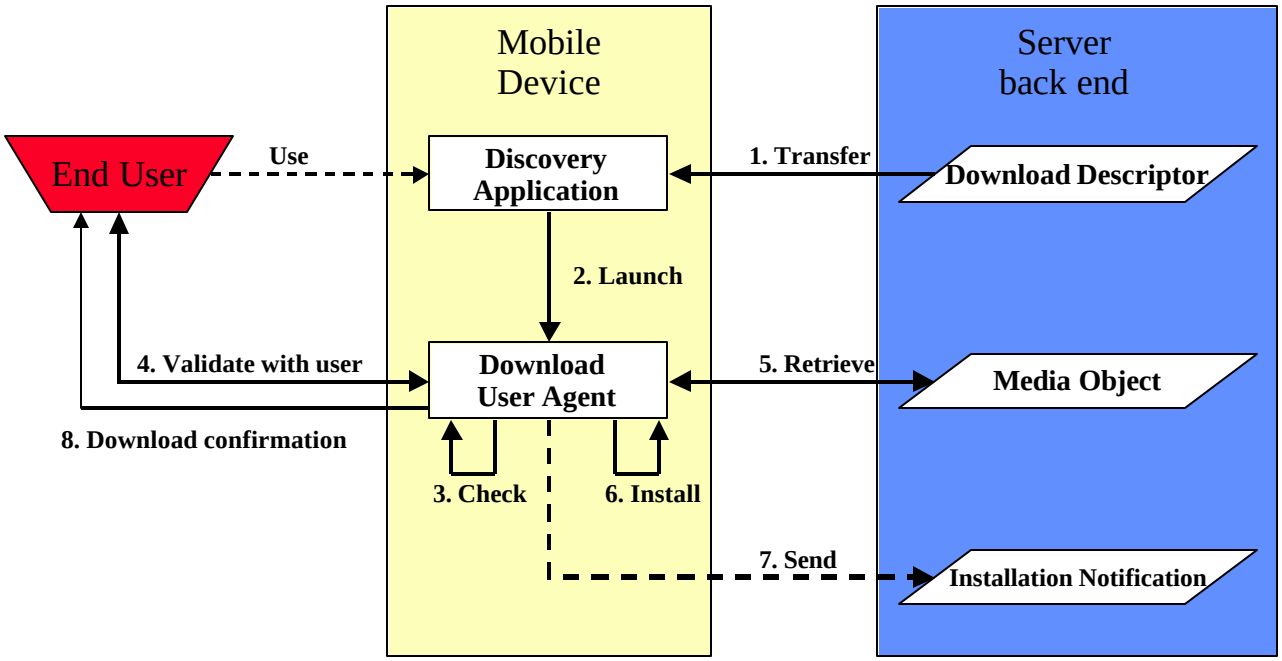

Figure 5. The picture describes the actions in the client device and the communication operations. These actions are referenced in the sections "Object Discovery Process" and "Object Installation Process".

5.1. 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 MUST at a minimum support either [W-HTTP] or [WSP], and MAY support either [WAPTLS] or [WTLS], as well as other protocols. 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 5.6.3 State Management of Download Transactions.

For details related to authentication see Section 5.6.2 Authentication of User.

5.1.1.1. Co-Delivery of Download Descriptor and Media Object

When the Download Descriptor is delivered to the device the server system may select to deliver the media object in a single delivery operation. This is described as a “Content Download transaction with co-delivery of Descriptor and media object” in Section 4.3.1 OMA DOWNLOAD. 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 object as the second part. The CID (Content ID) mechanism MUST be used in the Download Descriptor to reference the media object in a related multipart.

When the Media Object is Co-Delivered with the Download Descriptor the process continues as described in Section 5.2 Object Installation Process, with the exception that the Media Object is retrieved locally (in “step 5”) rather than from an external location. The same steps of “Launch of Download Agent”, “Capabilities check” and “User Confirmation” are still performed.

If the download transaction because of some reason is aborted before step 6 (or optionally during step 7) then the Media Object MUST be discarded from the device.

5.2. Object Installation process

Object installation is the process by which a Media Object is downloaded onto the device and made available for execution or rendering.

To install a Media Object, the Download Agent that is responsible for the processing of the Download Descriptor performs a number of actions, and SHOULD provide the user with appropriate feedback would one of the actions fail. It 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 download event, and MUST use the mechanism to report on a successful installation.

The Status Report is used to give the server entities a crude understanding of the reasons for an unsuccessful download operation. It is in many cases important feedback to a service to learn that a large number of devices reject the content, or that a large number of users abort the download when presented with a formal description of the media object.

The Status Report MUST be sent to the address defined in the InstallNotifyURI attribute. If the InstallNotifyURI attribute is missing from the Download Descriptor then no Status Report can be sent.

If the network service is lost during installation, an Error Code “Loss of Service” MUST be used in a Status Report if possible (it may be impossible to deliver the status report due to the network-service outage).

Table 1. Installation status code and associated message

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	Indicates that the media object does not match the 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.
951	Invalid DDVersion	Indicates that the device was not compatible with the “major” version of the Download Descriptor, as indicated in the attribute Version (that is a parameter to the attribute Media).
952	Device Aborted	Indicates that the device interrupted, or canceld, 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.

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 Descriptor is in error (e.g. a syntax error), an error message SHOULD be displayed to the user. If the InstallNotifyURL is defined then the Download Agent MUST post an “Invalid Descriptor” status report.

If the Version attribute of the Download Descriptor indicates a “major” version that is not supported by the client device it MUST 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 attributes are deployed. Thus, when processing the Download Descriptor, the following rules apply:

- Optional attributes, as defined in this specification, MUST be ignored if not supported.
- Unknown attributes MUST be ignored (these attributes MAY be defined outside this specification).
- If an attribute occurs more than once, all but the first occurrence MUST be ignored.
    - The only exception to this rule is the attribute Type that has well defined semantics when it occurs multiple times. If there are more occurrences of an attribute than the Download Agent can support then the Download Agent SHOULD accept as many attributes as it supports and after that ignore all subsequent attribute values as redundant (i.e. in case of multiple occurrences of an attribute the first one has the highest priority).

5.2.2. Step 3; Capabilities Check

The Download Agent MUST use the information in the descriptor, at a minimum size and type, to check whether the device is capable of using and/or rendering the media object. This is to prevent downloading of media objects that will not work properly.

If the device does not have sufficient memory for installing and storing the media object, the Downloading Agent SHOULD aid the user in reviewing memory usage and freeing sufficient memory for installation of the new media object. If there after this event is insufficient memory to store the media object on the device, the Download Agent MUST post an “Insufficient Memory” status report and notify the end user.

If the device based on the attributes in the Download Descriptor concludes that it lacks the capability, due to a reason different from insufficient memory, to perform a successful installation of the media object the Download Agent SHOULD notify the end user and MUST post a “Device Aborted” status report.

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

5.2.3. Step 4; User Confirmation

Using the information in the Download Descriptor the user SHOULD be given a chance to confirm that they want to install the media object.

The following information SHOULD, if available, be made available to the user:

- Name
- Vendor
- Size
- Type
- Description

The user SHOULD be prompted for confirmation if one or more types are not supported. If the user does not approve the downloading, the Download Agent MUST post an “User Cancelled” status report.

The Type attribute may occur multiple times in the Download Descriptor. It then indicates that the device is recommended to support all the listed media types in order to use the complete downloaded media object. In this case

the Download Agent should interpret the attributes in order of decreasing importance to the user, i.e. that first type attribute has the highest relevance for presentation to the end user (it would typically be the, one or more, most important media types to be rendered or executed). In case of use of packaging or wrappers the first, one or more, type attributes would represent the innermost media objects.

5.2.4. Step 5; Object retrieval

The retrieval of the media object is typically performed using HTTP (or HTTPS) but always according to the scheme in the ObjectURI attribute of the Download Descriptor. The Download Agent MUST support the HTTP scheme, and MAY support the HTTPS scheme. The Download Agent MUST at a minimum support either [W-HTTP] or [WSP], and MAY support either [WAPTLS] or [WTLS], as well as other optional protocols. For more details on specifics on the use of HTTP see section HTTP Specific Functionality.

The request for the 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 [W-HTTP] and [WSP].

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

If the connection with the network is lost during the retrieval of the media object the Download Agent MUST post a “Loss of Service” status report.

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

5.2.5. Step 6; Installation

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

The functionality of this step “Installation” depends on if the Installation Notification has been requested in the Download Descriptor or not.

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 object can now be rendered or executed.

Installation Notification requested

In case an Installation Notification has been requested in the Download Descriptor by means of the InstallNotifyURI attribute, then the installation process is split into two phases.

- The first phase (covered as step 6 in this specification) consists of a pre-installation where the device prepares the media object for rendering/execution to the largest extent possible without actually allowing it to be used.
- The second phase is dependant on the success of the next step (covered as step 7 in this specification), the execution of the Installation Notification. Only if that step is regarded as successful the media object will be made available for execution/rendering.

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

Installation is complete when the Media Object has been prepared for execution/rendering on the device, or an unrecoverable failure has occurred. In either case, the status MUST be reported (assuming the installation notification has been requested as indicated by the InstallNotifyURI) as described in the Section Status Report Functionality.

- If the installation succeeded the device MUST generate the status code “Success” in the Status Report as described in Section 5.2.6 “Step 7, Sending Installation Notification”.

- If the user canceled the downloading and installation before the installation completed (the sending of the installation notification was done) the device MUST generate the status code “User Cancelled” in the Status Report.
- If the installation of the software 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 software failed due to some other reason, independent of the end user, than lack of memory then the device MUST generate the status code “Device Aborted”
- If the Downloading User Agent immediately rejected the 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 the media object because the retrieved object conflicted in a non-recoverable way with the attributes 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 issued the Media Object MUST be discarded by the device. The logic of OMA Download is such that a failure remains a failure even if the status report could not be sent successfully. That is, from the server perspective a missing status report equals an error status code.

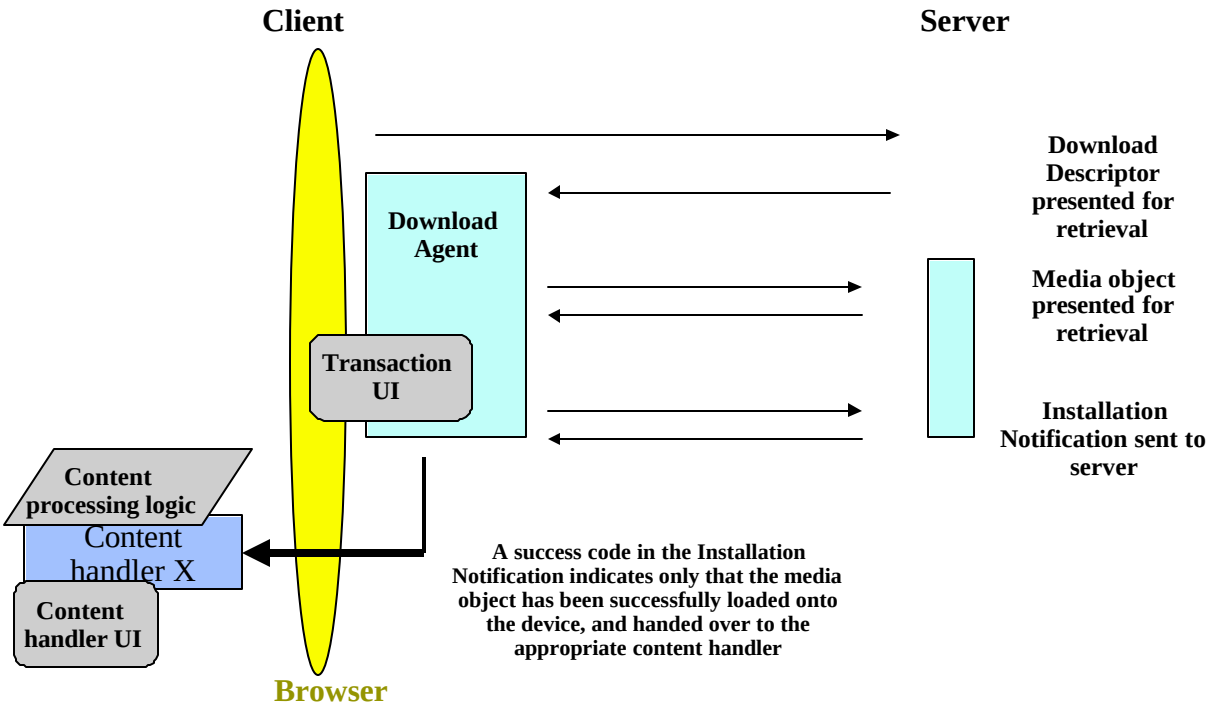

Figure 6. The successful completion of the download transaction is independent of the content handler that will finalize the processing of the retrieved media type. However, the Download Agent and the approriate content handler may negotiate during the transaction about the device capabilities to process the content.

5.2.5.1. Media Object installation parameter

The attribute installParam MAY be used to convey parameters to the Media Type specific Media Object Installer. If an installParam exists then the Download Agent MUST convey 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 does support the installation parameter, but there is an error in the parameter, then an error code (“attribute mismatch”) SHOULD be returned as defined in section 5.2.5.

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.

5.2.6. Step 7; Sending Installation Notification

This step, the sending of the Installation Notification, is valid only in the cases where it has been explicitly requested through the use of the attribute InstallNotifyURI in the Download Descriptor.

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

The installation status is reported by the use of the schema defined in the InstallNotifyURI attribute. The Download Agent MUST at a minimum support either [W-HTTP] or [WSP] protocols. If the scheme is HTTP then an HTTP POST request to the defined URL is performed. The HTTP schemeMUST be supported. However, if the transfer of the Media Object has been performed using a different scheme than HTTP then the Download Agent MUST be able to execute also 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 5.2 lists the valid status message codes and messages. The Section 5.3.1 defines the format of the installation notification.

The Media Object MUST NOT be released for use at the device unless the sending of the installation notification succeeds (in case the InstallNotifyURI has been defined in the Download Descriptor).

The sending of the Installation Notification is regarded as successful if the Download Agent receives a reply from the 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].

The exception to the logic defined above is the semantics of a “Well-Intentioned Attempt”. If the well-intentioned Installation Notification attempt brings no response from the server then the Download Agent MUST equal 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-Reply is implementation specific.

No content body should be returned in the HTTP reply to the device and, 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 unauthorised”, the reply prompts a modified retry).

If no well-intentioned attempt can be made then the device MUST NOT allow for the use of the media object. The device MUST indicate to the user that the download failed and remove the content from the device.

5.2.6.1. Installation Notification semantics

An important aspect of the Installation Notification mechanism in the Download Descriptor is to prevent a situation where the server has “knowledge” that the transaction completed, but the client perceives the transaction as failed (and

doesn't release the Media Object 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 implementation of the Download Agent in the device MUST make at least one well-intentioned attempt to send the Installation Notification to the address defined in the InstallNotifyURI attribute. 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 MUST immediately be removed from the device.

5.2.7. Step 8; Download Confirmation and next step

When the device has sent the status report, as described in step 7 above, or the transaction has completed without a Status Report (if no status report was requested in the download descriptor) 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 of either continue with a local context specific operation (often to render of execute the media object) or to continue with a browsing operation.

If the end user selects to continue with a browsing operation then the Download Agent SHOULD invoke the URL defined in the NextURL attribute. 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 attribute is missing from 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. In the case the end user selects to continue with a browsing operation the URL to be activated is implementation specific.

5.3. Status Report Functionality

The Status Report functionality in OMA Download covers reporting both successful (Installation Notification) as well as failed content download transactions.

The confirmation of a successful download operation (installation notification) 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. The server thus anyhow needs to have a robust logic to discard hanging transactions.

There are two major usage scenarios with respect to status reports:

- The Download Service wants installation notification as well as error codes if something fails in the transaction. This kind of functionality would typically be deployed in a pay per download environment (= FULL STATUS REPORTS).

- The Download Service wants neither error codes nor installation confirmation, and leverages only the metadata and capability negotiation features of the Download Descriptor (= NO STATUS REPORT).

The Status Report functionality described in the use cases above is implemented using the InstallNotifyURI attribute. If present, then the usage scenario FULL STATUS REPORTS is implemented as described above. If missing, then no status reports can be sent, and the usage scenario NO STATUS REPORT is implemented.

5.3.1. Status Report Formatting

The Status Report MUST contain the Status Code in a decimal format, followed by one or more <space> characters, followed by the textual representation of the error code (which MAY be followed by a <newline> characters).

5.4. 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 attribute 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.

5.5. Persistence of Download Descriptor attributes

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

5.6. HTTP Specific Functionality

5.6.1. Client capability advertisement

When a download operation over HTTP or HTTPS is to take place the device should advertise its capabilities (to the extent possible) by using the mechanism of HTTP request headers. Headers that SHOULD be included are Accept headers (at least Accept-Content) and User-Agent or UAProf.

The Server (or servers) supplying the Download Descriptor and the Media Object should use this information to select the best possible set of content for the device.

5.6.2. Authentication of user

Authentication of the user is not mandatory, but often a useful feature. Authentication MAY be performed at different levels of the protocol stack, but the HTTP basic authentication mechanism (as defined in RFC2617) MUST be supported by the client device.

If the server responds to the request for the Download Descriptor, or the Media Object, with a 401 (Unauthorised), the device MUST re-send the request (including potential cookies) with the user-supplied credentials in an Authorisation header field as specified in RFC2617. The credentials should be provided by the user—for example, a common mechanism would be to present a dialog to the user to enter a user name and password.

The network may also require proxy authentication. If a proxy responds to the request for the Download Descriptor, or the Media Object, with a 407 (Proxy Authentication Required), the device SHOULD re-send the request with the user-supplied credentials in a Proxy-Authorisation header field as specified in RFC2617.

5.6.3. 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. 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 Installation Notification. 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 installation notification (the Status Report).

The first method is based on the URL's that are exposed in the Download Descriptor. Each of the URL's may have a session identifier parameter that from the server point of view associates 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.

5.6.4. Transparency of Download Descriptor mechanism

The intention of the Download Descriptor download transaction mechanism is that it is 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 three 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 two other (optional) request-response operations SHOULD NOT be exposed to the Media Object specific content handler.

6. Download Descriptor

The Download Descriptor is a collection of attributes, used to describe a media object at a URI or URL (as defined in [RFC2396]). The defined attributes 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 attributes.

This section defines only the semantics of the Download Descriptor. The syntax is described in a separate section.

6.1. 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 attributes. 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 attributes (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 8 “XML Syntax for Download Descriptor”.

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

6.2. Download Descriptor attributes

The attributes in the descriptor MUST be formatted according to the syntax defined in the syntax section of this specification. 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, if that is supported, 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 attribute is defined using the following properties:

Name - The name of the attribute

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

Status - Whether the attribute 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 attribute

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

Comment - A remark concerning the application of the attribute

The attributes are defined in the following sections.

6.2.1. 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 attribute indicates the media type of the object to be executed or rendered. The type attribute 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 attribute MAY be different from the media type indicated in the HTTP header “content-type” as transport packaging MAY be used. SHOULD be used by the client to evaluate its capabilities relative to the content to be downloaded. The type attribute 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 attribute in the Download Descriptor.

6.2.2. size

Name	**`size`**
Definition	The number of bytes to be downloaded from the URI.
Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	Positive integer
Refinement	-
Comment	The storage size and the execution size are dependent on the environment and may be different from the value of the size attribute. The transport size may also be different, if compression or some packaging format is used. The size can be used to allocate sufficiently large data buffers for downloading in the client.

6.2.3. objectURI

Name	`objectURI`
Definition	The URI (usually URL) from which the media object can be loaded.

Status	Download Descriptor: Mandatory. User Agent: Mandatory
Datatype	URI
Refinement	-
Comment	The Download Agent MUST be able to use an HTTP GET to reference this URI in order to retrieve the media object.

6.2.4. installNotifyURI

Name	`installNotifyURI`
Definition	The URI (or usually URL) to which a 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
Refinement	-
Comment	The installNotifyURI attribute is the enabler of controlled transactions. If the attribute 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 1 (Section 5.2 Object Installation Process). If the attribute 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 URL. The URL is used both to report errors and process aborts, as well as to verify the successful installation of the media object.

6.2.5. 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: Optional
Datatype	A RFC2396 URL
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.

6.2.6. DDVersion

Name	`DDVersion`
Definition	The version of the Download Descriptor technology.
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	String (decimal)
Refinement	
Comment	The format of the Version is “major.minor”. A Download Agent not supporting the “major” version of the Download Descriptor ersion MUST send an “Invalid DDVersion” status code to the installNotifyURI and reject 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 “1.0”. The default DDVersion, when the attribute is omitted from the Download Descriptor, is “1.0”

6.2.7. name

Name	`name`
Definition	A user readable name of the Media Object that identifies the object to the user.
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 it. The Download Agent MAY also use the attribute Vendor to ensure uniqueness between names.

6.2.8. 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	SHOULD be displayed to the user before the download of the media object is accepted by the end user. Allows the user a last check before the transaction is completed. This may for example catch lacking synchronisation with regards to the presented content in the presentation server and the download server.

6.2.9. vendor

Name	`vendor`
Definition	The organisation that provides 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 attribute MAY be displayed to the user during installation. The attribute MAY be used by the Download Agent to create a unique name for the media object when it is stored in the device.

6.2.10. infoURL

Name	`infoURL`
Definition	A URL for further describing the media object
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URL
Refinement	Does not have any particular semantics, is intended for user interpretation.
Comment	The infoURL is used to for information that describes the media object rather than the download transaction.

6.2.11. iconURI

Name	`iconURI`
Definition	The URI of an icon
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	URI, URL

Refinement	-
Comment	May be used by the client to represent the media object (e.g. an application) in the user interface (e.g. application manager).

6.2.12. installParam

Name	`installParam`
Definition	An installation parameter associated with the downloaded media object
Status	Download Descriptor: Optional. User Agent: Optional
Datatype	VeryLongString
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.

6.3. Extensibility

The Download Descriptor used by OMA contains general metadata that is useful for all types of media. In some cases, however, the standardised attributes 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 extensibility is governed by a few basic rules:

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

6.3.1. 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 attributes that are specific to the installation of a specific media type. However, it is recommended that the installParam attribute be used for custom installation commands as described in section 5.2.5.1, 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 finalise 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. Relationship to Java™ MIDP OTA (informative)

The Download Descriptor for Generic Content Download may be used for all downloaded content types including MIDlets. However, whenever possible the Java™ Application Descriptor (JAD, as defined in [MIDPOTA]) should be used for Java™ MIDlet downloads.

The mechanism for Java™ MIDlet download is specified in the MIDP OTA Provisioning RP specification (see [MIDP] and [MIDPOTA]). This specification allows vendor specific enhancements to be defined. Some of the attributes defined in this specification may be used in a JAD file for download of MIDP objects.

7.1. MIDP OTA and OMA Download

The framework for generic content download currently consists of the OMA Download mechanism, and a media type specific content download mechanism for JAR files (MIDP download as defined in [MIDPOTA]). The OMA Download mechanism for generic content is not intended to be used in the application space of JAD.

- If the media type to be downloaded is a JAR file then the Download Descriptor should be a JAD file as defined in [MIDPOTA].
- If the media type to be downloaded is different from a JAR file, i.e. is not a MIDP object, then the Download Descriptor should be as defined in this specification.

The intent of the specification is to encourage similarity between MIDP download and OMA Download, However, no formal relationship between the two specifications exists (except that the generic content download references the MIDP specification for MIDP OTA Provisioning).

8. 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.dd+xml has been registered with IANA for use as the Download Descriptor. The XML Schema [XMLSchema1] [XMLSchema2] for this media type is defined in section 8.2.

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.

8.1. Example

```
<media xmlns="http://www.openmobilealliance.org/xmlns/dd">
      <type>image/gif</type>
      <objectURI>http:/download.example.com/image.gif</objectURI>
      <size>100</size>
      <installNotifyURI>http:/download.example.com/
            image.gif?id=image</installNotifyURI>
</media>
```

8.2. XML Schema

```
<xsd:schema
      targetNamespace="http://www.openmobilealliance.org/xmlns/dd"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:dd="http://www.openmobilealliance.org/xmlns/dd"  >

<xsd:simpleType name="ShortString">
      <xsd:restriction base="xsd:string">
             <xsd:maxLength value="40" />
      </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="LongString">
      <xsd:restriction base="xsd:string">
            <xsd:maxLength value="160" />
      </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="VeryLongString">
      <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
      </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="URI">
      <xsd:restriction base="xsd:anyURI">
            <xsd:maxLength value="128" />
      </xsd:restriction>
</xsd:simpleType>

<xsd:element name="name" type="dd:ShortString" />
<xsd:element name="DDVersion" type="dd:ShortString" />
<xsd:element name="objectURI" type="dd:URI" />
```

```
<xsd:element name="size" type="xsd:positiveInteger" />
<xsd:element name="type" type="dd:ShortString" />
<xsd:element name="vendor" type="dd:ShortString" />
<xsd:element name="description" type="dd:LongString"  />
<xsd:element name="installNotifyURI" type="dd:URI"  />
<xsd:element name="nextURL" type="dd:URI" />
<xsd:element name="infoURL" type="dd:URI" />
<xsd:element name="iconURI" type="dd:URI" />
<xsd:element name="installParam" type="dd:VeryLongString" />

<xsd:element name="media">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element ref="dd:objectURI" />
                  <xsd:element ref="dd:size"  />
                  <xsd:element ref="dd:type" maxOccurs="unbounded"/>
                  <xsd:element ref="dd:name" minOccurs="0"/>
                  <xsd:element ref="dd:DDVersion" minOccurs="0"/>
                  <xsd:element ref="dd:vendor" minOccurs="0" />
                  <xsd:element ref="dd:description" minOccurs="0" />
                  <xsd:element ref="dd:installNotifyURI" minOccurs="0" />
                  <xsd:element ref="dd:nextURL" minOccurs="0" />
                  <xsd:element ref="dd:infoURL" minOccurs="0" />
                  <xsd:element ref="dd:iconURI" minOccurs="0" />
                  <xsd:element ref="dd:installParam" minOccurs="0" />
            </xsd:sequence>
      </xsd:complexType>
</xsd:element>

</xsd:schema>
```

Appendix A. Example of Download Transaction (informative)

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

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.dd+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.wapforum.org/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>
```

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.dd+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: 50
Content-Type: application/vnd.oma.dd+xml; charset=utf-8

<media xmlns="http://www.openmobilealliance.org/xmlns/dd">
  <type>image/gif</type>
  <objectURI>http:/foo.bar.com/pic-dir/picture.gif
               </objectURI>
  <size>1234</size>
  <installNotify-URI>http:/foo.bar.com/status</installNotify-URI>
```

```
</media>
```

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…
```

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
```

The response from the server might be:

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

Appendix B. Static Conformance Requirements (Normative)

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

Item	Function	Reference	Status	Requirement
DL-OTA-C-001	Support for separate delivery of download descriptor and media object	5	M	
DL-OTA-C-002	Support for co-delivery of download descriptor and media object	5.1.1.1	O	DL-OTA-C-003
DL-OTA-C-003	Support multipart/related or application/vnd.wap.multipart.related	5.1.1.1	O	
DL-OTA-C-004	Support for transfer protocol	5.1.1	M	DL-OTA-C-005 OR DL-OTA-C-006
DL-OTA-C-005	Support for W-HTTP transfer protocol	5.1.1	O	W-HTTP:MCF AND DL-OTA-C-022
DL-OTA-C-006	Support for WSP	5.1.1	O	WSP:MCF
DL-OTA-C-007	Support for WAPTLS or WTLS for secure connections	5.1.1	O	WAPTLS:MCF OR WTLS:MCF
DL-OTA-C-008	Discard media object if download transaction aborted.	5.1.1.1	M	
DL-OTA-C-009	Discard download descriptor if major version not supported.	5.2.1	M	
DL-OTA-C-010	Correct processing of download descriptor	5.2.1.1	M	
DL-OTA-C-011	Use download descriptor to do capability checking	5.2.2	M	
DL-OTA-C-012	Allow user to confirm media object download	5.2.3	O	
DL-OTA-C-013	Use objectURI to retrieve media object	5.2.4	M	
DL-OTA-C-014	Send status report to indicate success or failure if InstallNotifyURI specified	5.2.5	M	
DL-OTA-C-015	Discard media object if error status report sent	5.2.5	M	
DL-OTA-C-016	Existence of Media Object Installer that supports installation parameter	5.2.5.1	O	DL-OTA-C-017
DL-OTA-C-017	Pass installParam value to Media Object Installer	5.2.5.1	O	
DL-OTA-C-018	Prevent access to media object until status report succeeds (at least one well-intentioned attempt).	5.2.6	M	
DL-OTA-C-019	Support same scheme for status report as used for media object download	5.2.6	M	

Item	Function	Reference	Status	Requirement
DL-OTA-C-020	Navigate browser to nextURL	5.2.7	O	WAESpec:MCF
DL-OTA-C-021	Proper formatting of status report	5.3.1	M	
DL-OTA-C-022	Support for HTTP Basic Authentication	5.6.2	O	
DL-OTA-C-023	Use media type delared by transport or packaging to identify download descriptor	6.1	M	

Item	Function	Reference	Status	Requirement
DL-DD-C-001	Ignore unsupported optional attributes	5.2.1.1	M	
DL-DD-C-002	Ignore unknown attributes	5.2.1.1	M	
DL-DD-C-003	Ignore multiple instances of attribute (except Type)	5.2.1.1	M	
DL-DD-C-004	Ignore unknown attribute values	6.3	M	

Appendix C. Change History (Informative)

Type of Change	Date	Section	Description
Class 0	12-Sept-2002		The initial version of this document.
Class 2	05 Dec 2002		Fix references to HTTP/S and rework SCR table
Class 2	21-Feb-2003	8.2	Modifications to XML Schema

Appendix D. Media Type Registration

```
...Registration of MIME media type application/vnd.oma.dd+xml

   MIME media type name:               application
   MIME subtype name:                  vnd.oma.dd+xml
   Required parameters:                none
   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="2.1" indicates version 2.1.

  Encoding considerations: See [XMLMIME].

  Security considerations: See [XMLMIME].

  Interoperability considerations:
  The OMA Download specifications
  [OMADL] specify user agent (Download Agent) conformance rules that
  dictate behaviour that must be followed when dealing with, among other
  things, unrecognized elements.

  Published specification:
  The OMA Download specification is published at
  http://www.openmobilealliance.org/

  Applications which use this media type:
        OMA Download agents, see [OMADL].

  Additional information:

     Magic number: There is no single initial byte sequence that is always
     present for Download Descriptor files.

     File extension: .xml or .dd

     Macintosh File Type code: TEXT

   Person & email address to contact for further information: Open Mobile Alliance
    <technical-comments@mail.wapforum.org>

   Intended usage: COMMON

   Author/Change controller: The OMA Download specifications are a work
   product of the Open Mobile Alliance's WAG Working Group.  The Open Mobile Alliance has
   change control over these specifications.

....Fragment identifiers

        Fragment identifiers are not used for this media type.

....References

[OMADL] "OMA Download OTA Specification", Open Mobile Alliance
Specification. Available at <http://www.openmobilealliance.org/>.

[XMLMIME] Murata, M., St.Laurent, S., Kohn, D., "XML Media Types", RFC
3023, January 2001.
```