Download Architecture

Approved Version 1.0 – 25 Jun 2004

Open Mobile Alliance

OMA-Download-ARCH-V1_0-20040625-A

Contents

1. SCOPE (INFORMATIVE) 4
2. REFERENCES 5
3. TERMINOLOGY AND CONVENTIONS 6
3.1 CONVENTIONS 6
3.2 DEFINITIONS 6
3.3 ABBREVIATIONS 7
4. INTRODUCTION (INFORMATIVE) 8
4.1 OMA DOWNLOAD 8
4.1.1 Media Type Specific Extensions 8
4.1.2 Benefits of OMA Download 8
4.2 OMA DIGITAL RIGHTS MANAGEMENT 9
4.3 THE OMA DOWNLOAD DOCUMENTS 10
5. SYSTEM OVERVIEW 11
5.1 DATA MODEL 11
5.2 NETWORK ENTITIES 12
6. DOWNLOAD AND DRM USE CASES 13
6.1 BASIC DOWNLOAD USE CASE 13
6.1.1 Use Case DL-1: HTTP Download 14
6.2 OMA DOWNLOAD USE CASES 14
6.2.1 Use Case DL-2: Combined Descriptor and Media Object 14
6.2.2 Use Case DL-3: Separate Descriptor and Media Object 15
6.3 DRM USE CASES 15
6.3.1 Use Case DRM-1: Forward-lock 16
6.3.2 Use Case DRM-2: Combined Delivery 16
6.3.3 Use Case DRM-3: Separate Delivery (Push) 16
APPENDIX A. HTTP AND MIDP DOWNLOAD (INFORMATIVE) 18
A.1 BASIC HTTP DOWNLOAD 18
A.1.1 Using HTTP for Download 18
A.1.2 When is HTTP Insufficient? 19
A.2 JAVA™ MIDLET DOWNLOAD 19
APPENDIX B. CHANGE HISTORY (INFORMATIVE) 21
B.1 APPROVED VERSION HISTORY 21
B.2 DRAFT/CANDIDATE VERSION 1.0 HISTORY 21

1. Scope (Informative)

The scope of OMA Download is procedures for making the downloading of media objects from the Web easy and reliable, and possible to charge for. Also in the scope of OMA Download is the ability to preview media objects and to prevent downloaded content from being forwarded (copied) to other users. Preview and copy protection are part of Digital Rights Management (DRM). A complete DRM technology is, however, **not** in scope of OMA Download. These basic DRM features - preview and copy protection - are included because of the urgency to deploy a standardised solution.

2. References

[WAP221]	"Specification of WAP Conformance Requirements". WAP Forum™. WAP-221-CREQ. http://www.openmobilealliance.org/
[MIDP]	"Over The Air User Initiated Provisioning Recommended Practice", 2001, SUN Microsystems, Inc.
[MSERVICE]	"Download Requirements for M-Service", 30 November 2001. Liaison Statement from the M-Service Interest Group of the GSM Association to the WAP Forum.
[RFC2046]	"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", http://www.ietf.org/rfc/rfc2046.txt
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC2246]	"The TLS Protocol", http://www.ietf.org/rfc/rfc2246.txt
[RFC2396]	"URI", http://www.ietf.org/rfc/rfc2396.txt
[RFC2616]	"Hypertext Transfer Protocol - HTTP/1.1", http://www.ietf.org/rfc/rfc2616.txt
[RFC2617]	"HTTP Authentication: Basic and Digest Access Authentication", http://www.ietf.org/rfc/rfc2617.txt
[RFC2965]	"HTTP State Management - Cookies", http://www.ietf.org/rfc/rfc2965.txt
[WAP248]	"User Agent Profiles", WAP Forum™, WAP-248-UAPROF, http://www.openmobilealliance.org/
[OTA]	"Download OTA", Open Mobile Alliance™, OMA-Download-OTA-v1_0, http://www.openmobilealliance.org/
[DRM]	"Digital Rights Management", Open Mobile Alliance™, OMA-Download-DRM-v1_0, http://www.openmobilealliance.org/
[DRMREL]	"DRM Rights Expression Language", Open Mobile Alliance™, OMA-Download-DRMREL-v1_0, http://www.openmobilealliance.org/
[DRMCF]	"DRM Content Format", Open Mobile Alliance™, OMA-Download-DRMCF-v1_0, http://www.openmobilealliance.org/
[WAP210]	"WAP Architecture", WAP Forum™, WAP-210-WAPArch, http//www.openmobilealliance.org/
[WAP223]	"HTTP State Management", WAP Forum™, WAP-223-HTTPSM, http://www.openmobilealliance.org/
[WAP230]	"Wireless Session Protocol - WSP", WAP Forum™, WAP-230-WSP, http://www.openmobilealliance.org/
[WAP298]	"XHTML Mobile Profile", WAP Forum™, WAP-298-XHTML, http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

All sections in this document are informative.

3.2 Definitions

Discovery application
A user agent in the device that discovers media objects on behalf of the user.

Download agent
A user agent in the device responsible for downloading a *media object* described by a *download descriptor*.

Download descriptor
Metadata about a *media object* and instructions to the *download agent* for how to download it.

Download server
A Web server hosting *media objects* available for download. May be the same physical server as the *presentation server*.

DRM Agent
A user agent in the device that enforces DRM *rights* for media objects on the device.

DRM Packager
A program that packages *media objects* into DRM format.

DRM Message
A message containing an optional rights object and a media object.

Media object
A resource on a Web server that can be downloaded.

Note: Background pictures, ring tones, themes, games, music, and audio and video streams are collectively referred to as *media objects*.

Media type
A MIME media type [RFC2046].

MIDlet Download
Refers to the Java 2 Micro Edition (J2ME) Mobile Internet Device Profile Over-the-Air User Initiated Provisioning Recommended Practises. This defines the method of downloading and managing the lifecycle of J2ME applications, referred to as MIDlets.

Presentation server
A Web server presenting a download service to the user. May be the same physical server as the *download server*.

Note: This may be a Web "portal" or a Web site dedicated to mobile downloading, for example.

Rights
Permissions and constraints defining under which circumstances access is granted a media object.

Rights object
An instance of a rights language document, with one or many rights expressions.

Status report server

A Web server accepting status reports from the *download agent*.

3.3 Abbreviations

BLOB	Binary Large Object
DRM	Digital Rights Management
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
MIDlet	Java™ MIDP program
MIDP	Mobile Information Device Profile
MIME	Multipurpose Internet Mail Extensions
MMS	Multi-Media Messaging Service
OMA	Open Mobile Alliance
SMS	Short Message Service
WAP	Wireless Application Protocol
WML	Wireless Markup Language
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language

4. Introduction (Informative)

OMA Download consists of two broad categories of functionality:

- **Download -** to download *media objects* to a device.
- **Digital Rights Management (DRM) -** to control the usage of *media objects* on the device.

The following sections describe Download and DRM in more detail. The final section provides an overview of the OMA Download specification suite.

4.1 OMA Download

OMA Download is based on two already existing and successful download services: Basic HTTP Download and MIDlet Download. These services are described in Appendix A.

The purpose of OMA Download is to provide a service similar to MIDlet Download. The difference between MIDlet Download and OMA Download is that OMA Download is not designed specifically for the downloading of Java™ MIDlets, or any other specific media type - OMA Download is a general download framework. The similarities between the two download services permit re-use of server infrastructure and, in the device, to the extent possible, make the download of general *media objects* not significantly different from the download of a new MIDlet. Consistency is a basis for a good user experience. OMA Download does not, however, imitate MIDlet Download exactly in all technical details. But to the users downloading media objects to the device and to the content providers publishing media objects, MIDlet Download and OMA Download are very similar.

OMA Download can download any type of *media object*. The typical *media object* targeted by OMA Download is downloaded and stored persistently in the device, in order to personalise the device or enhance its functionality. Ring-tones, background images, and Java™ MIDlets are *media types* used for this purpose.

OMA Download, just like MIDlet Download, extends Basic HTTP Download with two additional steps:

1. **Before download**, a description of the media object is downloaded. The description is a small file, the *download descriptor*, containing metadata about the *media object* and instructions to the *download agent* in the device, how to download the *media object*.

   The *media object* is downloaded using standard Basic HTTP Download.

2. **After download**, a status report is posted to a Web site, indicating the outcome of the download. The user can be directed to a Web location provided in the *download descriptor*.

Some *media types* have specific requirements on the downloading procedure. To download a description of a music-stream (that can be used to trigger a streaming session) requires specific functionality not required to download a ring-tone. For this reason, it is possible to extend the general download framework with *media type* specific behaviour.

4.1.1 Media Type Specific Extensions

The *download descriptor* contains general metadata that is useful for all types of media. In some cases - such as MIDlets and media streams -the general metadata is 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 download procedure.

An example of how the *download descriptor* can be extended is available in [OTA].

4.1.2 Benefits of OMA Download

For the content provider and operator, the main benefits of OMA Download over standard Basic HTTP Download are:

- The download is confirmed, with the status report posted to the network after the download. It can be used for monitoring the quality of the service, and as the basis for billing.
- A description of the downloadable *media objects* can be published on the *presentation server* in a format independent of the presentation language - XHTML, WML, HTML etc. - supported by the devices downloading the *media objects*. The content provider creating the *media objects* just needs to publish the *download descriptors* on its Web site, the front-end of the download service can be provided by someone else; for example, a special download Web site with *media objects* from many different content providers.
- The user can be directed to a Web location provided in the *download descriptor*.

For the user, the main benefits of OMA Download are (depending on to what extent the device takes advantage of the OMA Download features):

- The probability of downloading a media object that the device cannot support is minimised, capability checks are made in the device just before the download starts.. The user is prevented from downloading things that do not work well in the device.
- The user is given a chance to confirm the download based on information in the *download descriptor* and the device's capabilities; "How much room do I have left for other objects after having downloaded this one?"
- The *download descriptor* is the basis for a familiar "download user interface" in the device, a sequence of user interactions that are independent of Web sites and networks.
- The *media object* metadata in the *download descriptor* can be stored in the device and be made available in the user interface as "properties" (name, vendor, type, Info-URL, etc.) of the *media object*.

And finally, the concept of a *descriptor* containing metadata about media objects, Web resources, is extensible to other use cases than download.

4.2 OMA Digital Rights Management

There is a need for content providers and operators to control the usage of downloaded *media objects*. Download is the means by which media object is delivered to the device, whereas Digital Rights Management (DRM) is the means to control the usage of *media object* once it has been downloaded.

DRM enables content providers to define *rights* for *media objects*. It is possible to associate different *rights* with a single *media object*. Different *rights* may have different prices. A content provider can grant a user the *rights* to preview *media objects* for free and charge the user only for the full usage *rights*. DRM makes it possible to sell the *rights* to use the *media object*, rather than selling the *media objects* itself. The *rights* are expressed using a rights language that is an XML application. An instance of the rights language, expressed as an XML document, is called a *rights object*. When a *media object* is distributed in encrypted form, the key is with the *rights object*. Thus the *rights* are required in order to use the *media object*.

Here are some examples of *rights* supported by OMA DRM:

- The right to play, display, or execute a *media object*.
- The right to play, display, or execute a *media object* a limited number of times. (By setting the number of times an object can be used to one, a "preview" right is defined.)

OMA DRM uses the following DRM components, found in most DRM systems:

- **Rights Expression Language**. Used to express *rights* and constraints governing the usage of the *media object* on the device.

- **Content format.** For usage in the DRM system. This comprises encryption of *media objects* using keys and a specified algorithm. Where no protection is deemed necessary, *media objects* may also be packaged as plaintext. A tool, the *DRM packager*, is used on the server to package media objects into the DRM format.
- **Metadata**. Data about the media object used to: (a) identify content (a URI [RFC2396]); and (b) help the user understand what the media object actually is (descriptive metadata). Media objects become part of the DRM system by optionally encrypting and packaging the media object into a DRM package.

OMA DRM has a low level of application level security. A secure DRM technology, including strong trust between the devices and content providers, is not in the scope of OMA DRM. Transport level security (WTLS and TLS) can be used to secure the transport of media objects between the user's device and a server. In DRM, however, since the user is a potential attacker – making illegal copies, etc. – transport security alone is not sufficient.

4.3 The OMA Download Documents

This document, the OMA Download Architecture document, specifies the overall architecture of OMA Download, the components from which the system is built and the interaction amongst those components. Protocols and formats are not specified in this document. Here is an overview of the OMA Download suite of specifications.

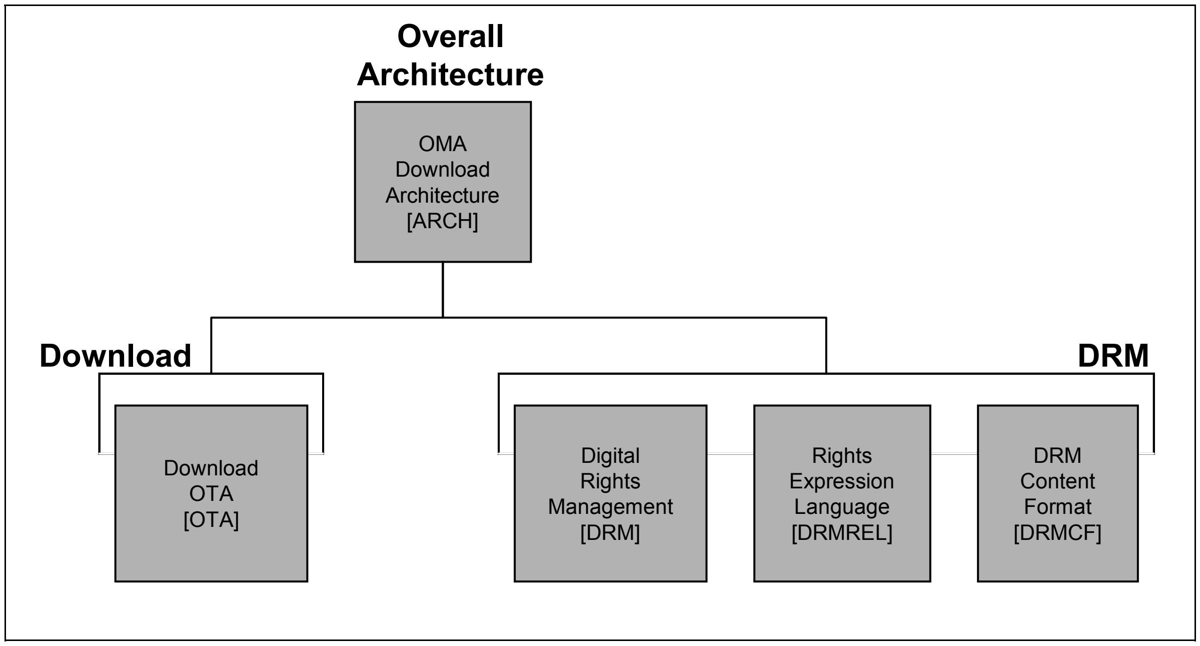

Overall Architecture
OMA Download Architecture [ARCH]
Download
DRM
Download OTA [OTA]
Digital Rights Management [DRM]
Rights Expression Language [DRMREL]
DRM Content Format [DRMCF]

Figure 1 Documents Overview.

The other documents of the OMA Download specification suite are:

Download OTA [OTA]
The procedure for downloading content using a download descriptor, and the definition of the download descriptor.

Digital Rights Management [DRM]
An extension to the downloading procedure, downloading of protected media objects.

Rights Expression Language [DRMREL]
The DRM rights language used by OMA.

DRM Content Format [DRMCF]
Content formats for transporting protected media objects.

5. System Overview

This section defines the architecture of the OMA Download system, including the DRM parts. It defines logical entities in the network and the device that are part of the system. The interaction between the entities, application level protocols and data formats, is defined elsewhere in the OMA Download specifications [OTA] [DRM] [DRMREL] [DRMCF].

5.1 Data Model

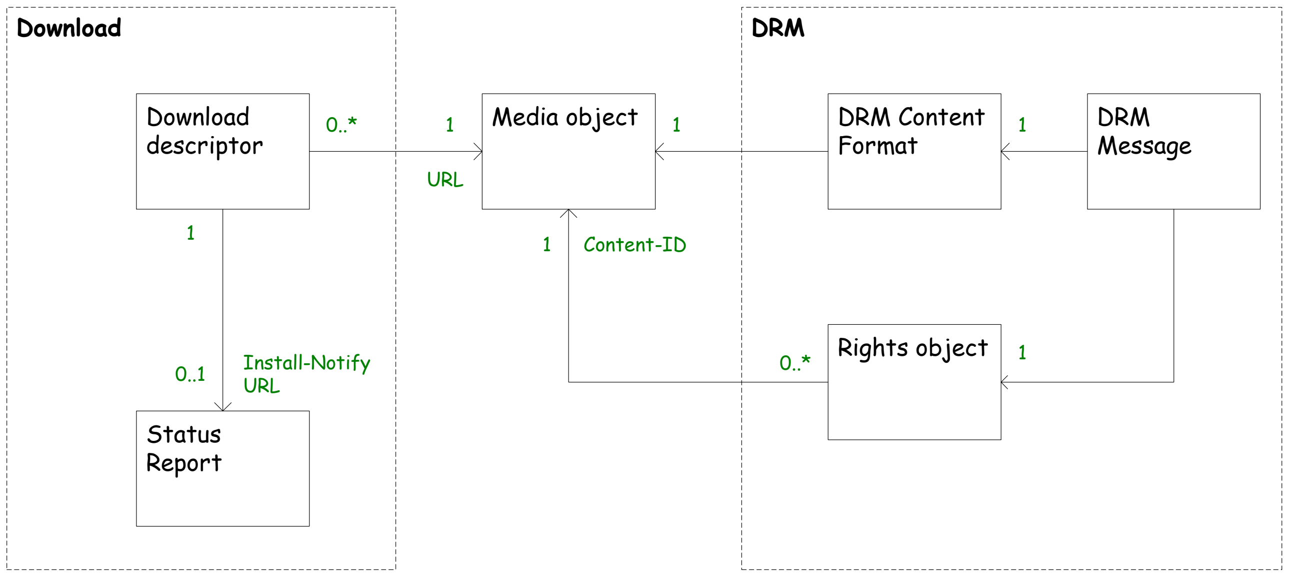
This section defines the types of data used by the OMA Download system and the relationships between the data.

From the perspective of the OMA Download system, a *media object* is nothing but an opaque object, a BLOB. There are, however, some basic requirements on *media objects*. A downloadable *media object* is located on the Web using a URL (e.g. "http://example.org/image001.gif") and is identified using a URI (e.g. "doi:10.1444/example-02-02" or "http://example.org/image001.gif").

The *download descriptor* contains the following: Metadata about the media object; exactly one URL of the *media object* to be downloaded; and zero or one URL (the **installNotify** URI) to which a *status report* should be posted. The *download descriptor* must know exactly what media object it is associated with, that is why it contains the URL of the *media object* (it also explains the arrow on the association in the diagram). If the *media object* is moved to another URL, the *download descriptor* must be updated. The *media object* itself, however, has no knowledge of what *download descriptor* it is associated with.

The *DRM content format* contains protected *media objects*. The *rights object* contain exactly one reference (the **Content-ID** URI) to the *media object* for which the *rights* apply. The content ID should be globally unique.

The *DRM Message* is used to encapsulate *rights object* and a *DRM content format* in one entity.



In the diagram, multiplicity is indicated with "0", "1", and "*" and direction is indicated with an arrow. In the relationship between the *download descriptor* and the *media object*, there can be exactly one ("1") *media object* for each descriptor, but zero or many ("0..*") *download descriptors* for each object. The direction of the relationship between the *download descriptor* and the *media object* indicates that the *download descriptor* must have a "pointer" (URL) to the *media object*.

5.2 Network Entities

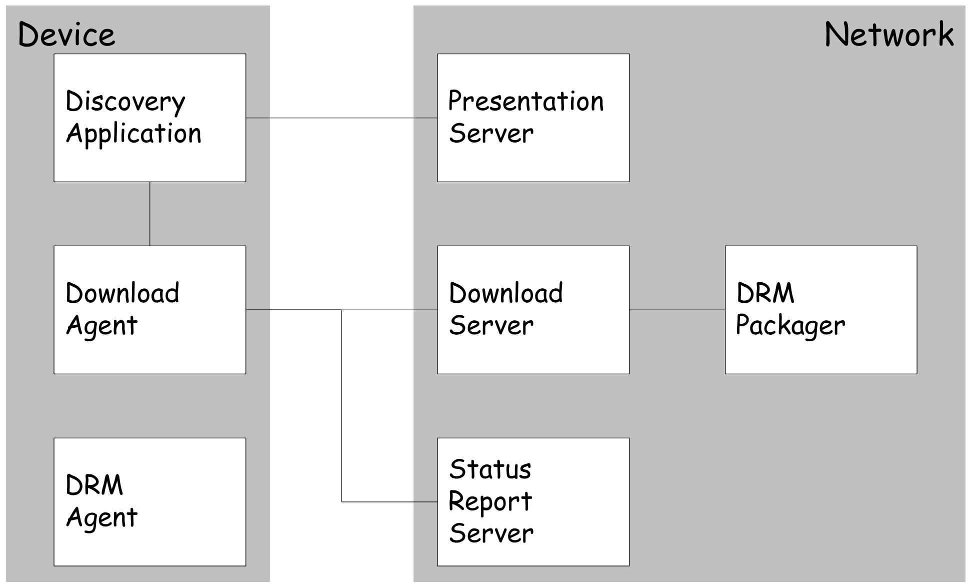
This section defines the network entities of the OMA Download system. These are logical entities and need not represent physical network nodes (servers, etc).

The User discovers *media objects* on the Web by using a Web browser or an application specifically created for one type of media. 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. And 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*.

The *presentation server* may be a Web "portal" operated by the operator or a third party. The *download server* is responsible for the download transaction, moving the actual *media object* from the server to the *download agent*. The *download descriptor* can be located on either the *presentation server* or the *download server*. When the *discovery application* downloads the *download descriptor* it launches the *download agent* that takes over the remaining part of the downloading. If the *media object* is protected it has been wrapped into a *DRM Container* by the *DRM packager*. The status report is posted to the *Status Report Server*. The *DRM packager* is also responsible for generating the *rights*.

> **Note**: The *presentation server, download server,* and *status report server* may be one and the same network node, or separate nodes.

In the device, the *DRM Agent* handles DRM *media object*.



To see the interaction between the entities, the use cases in section 6 are useful.

6. Download and DRM Use Cases

This section describes the key use cases of OMA Download and DRM.

The use cases are summarized in the following two tables. Each use case supports a set of features.

The download use cases are summarised in the following table.

Feature Use case	**Present object metadata to user before download starts**	**Device checks object size and type before download**	**User confirms before download starts**	**Download from standard HTTP server**	**After download, post status report to specified URL**	**After download, redirect user to specified URL**
DL-1: HTTP Download	Yes. Content in Web page.	Yes Based on XHTML link attributes[1].	Yes. Depending on Web page design.	Yes	No.	No.
DL-2: OMA Download, combined descriptor and object	Yes. Data in *download descriptor*.	Yes. Based on XHTML link attributes.	No. Unless as in DL-1.	Yes	Yes.	Yes.
DL-3: OMA Download, separate descriptor and object	Yes. Data in *download descriptor*.	Yes Based *download descriptor*	Yes.	Yes	Yes.	Yes.

The DRM use cases are summarised in the following table:

Feature Use case	**Specify that an object may not be copied from one device to another**	**Specify usage rights for an object**	**Distribute encrypted objects**	**Download an object with its usage rights**	**Download an encrypted object, get usage rights and key separately**
DRM-1: Forward-lock	Yes	No	No	No	No
DRM-2: Combined delivery	Yes	Yes	No	Yes	No
DRM-3: Separate delivery	Yes	Yes	Yes	No	Yes

The OMA Download use cases are specified at low level in [OTA] and the DRM use cases in [DRM].

6.1 Basic Download Use Case

The Basic Download use case is a standard HTTP transaction, exactly as used on the Web today. It is not defined by OMA Download, but documented here because it is used by the OMA Download use cases.

[1] A relevant XHTML link attribute [WAP280] is “type”, to indicate the media type of the link target. For example:

```
<a href=”anObject.ext” type=”image/gif”>Download image</a>
```

6.1.1 Use Case DL-1: HTTP Download

User initiates download of a *media object* from an HTTP server (on the Web) to an HTTP client (in the device) by using one HTTP GET request and response.

When this use case starts, the *media object* to be downloaded must be available on an “http” URL and the URL must also be available in the device to the User (e.g. as a link in a Web page).

When this case ends, if the main scenario is completed, the *media object* is available in the device.

Main scenario:

1. User initiates a GET request to the URL - by selecting a link in a Web page, for example.
2. Client sends GET request to the server and waits for a response.
3. Server serves up the requested resource, the *media object*, and sends GET response back to Client.
4. Client accepts GET response, with the HTTP headers and the data, from the server.

Extensions:

3a. Server uses UAProf or the request headers (e.g. “Accept”, “User-agent”, “UAProf”) to select an appropriate variant of the response.

4a. If the media type (in the “Content-type” header) is not supported, the Client closes the network connection and the server times-out.

4b. If the size (in the “Content-length” header) is too big, the Client closes the network connection and the server times-out.

More information about usage of standard HTTP for download is available in Appendix A.

6.2 OMA Download Use Cases

The OMA Download use cases extends the Basic HTTP Download use case by using a *download descriptor*. For the sake of brevity, only the extensions to the Basic HTTP Download use case are indicated in the following OMA Download use cases.

6.2.1 Use Case DL-2: Combined Descriptor and Media Object

A *media object* and a *download descriptor* is downloaded from an HTTP server by using one GET request and response. Because the *media object* and the *download descriptor* are both delivered together, the user is unable to confirm the download before the delivery of the *media object*. Optionally, a status report is posted to a URL specified in the *download descriptor*.

When this use case starts, the *media object* and the corresponding *download descriptor* is packaged into one multipart entity available on an “http” URL, the URL must also be available in the device to the User (e.g. as a link in a Web page).

When this case ends, if the main scenario is completed, the *media object* is available in the device.

Main scenario:

1. User initiates a GET request to the URL - by selecting a link in a Web page, for example.
2. Client sends GET request to the server and waits for a response.

3. Server serves up the requested resource, the multipart with the *media object* and the *download descriptor*, and sends GET response back to Client.
4. Client accepts GET response, with the HTTP headers and the data, from the server.
5. The information in the *download descriptor* is presented to the User.
6. The *media object* is made available to the User (e.g. saved on the filesystem).

Extensions:5b. If the *download descriptor* indicates that a status report shall be posted, a status report is posted to the specified URL.

6.2.2 Use Case DL-3: Separate Descriptor and Media Object

A *media object* and a *download descriptor* is downloaded from an HTTP server by using two GET request and response. The user is able to confirm the download based on pre-download capability checks in the device, and via a device specific download user interface. Optionally, a status report is posted to a URL specified in the *download descriptor*.When this use case starts, the *media object* and *download descriptor* are available on two separate "http" URLs, the *download descriptor* URL must also be available in the device to the User (e.g. as a link in a Web page). The *media object* URL is available in the *download descriptor*.

When this case ends, if the main scenario is completed, the *media object* is available in the device.

Main scenario:

1. User initiates a GET request to the URL - by selecting a link in a Web page, for example.
2. Client sends GET request to the server and waits for a response.
3. Server serves up the requested resource, the *download descriptor*, and sends GET response back to Client.
4. Client accepts GET response, with the HTTP headers and the data, from the server.
5. The information in the *download descriptor* is analysed by the device (capability checks) and User is given a chance to confirm the download.
6. HTTP Download is used to deliver the media object.
7. The *media object* is made available to the User (e.g. saved on the filesystem).

Extensions:

6b. If the *download descriptor* indicates that a status report shall be posted, a status report is posted to the specified URL.

6.3 DRM Use Cases

The DRM use cases are an extension to the OMA Download use cases. The idea in OMA Download is to deliver bits to the device, regardless whether the bits are DRM protected or not. DRM is independent of the download; it governs the control of the *media object* once it has been delivered to the device.

The DRM use case adds one step before download: the DRM packaging of the *media object*, statically or dynamically. And one step after the media object has been downloaded to the device; enforcement of the DRM usage *rights*.

6.3.1 Use Case DRM-1: Forward-lock

A *media object* is marked as "forward-lock" and published on an HTTP server (on the Web), from which an HTTP client (in the device) by using one HTTP GET request and response downloads it.

When this case ends, if the main scenario is completed, the *media object* is available in the device. It can be used, but it cannot be copied to another device.

Main scenario:

1. The Content Provider, using a *DRM Packager*, packages the *media object* into one *DRM* Messageand makes the URL available to the device (e.g. by publishing it on a Web page).
2. HTTP Download or OMA Download is used to deliver the *DRM Message*.
3. The User is using the *media object*. The DRM agent ensures that the *media object* is not copied to another device, via IrDA or Bluetooth, or sent from the device in an MMS or email.

Extensions:

None.

6.3.2 Use Case DRM-2: Combined Delivery

A protected *media object*, together with the corresponding *rights object*, is downloaded from an HTTP server.

When this case ends, if the main scenario is completed, the *media object* is available in the device. It can be used only according to the granted usage rights.

Main scenario:

1. The Content Provider, using a *DRM Packager*, packages the *media object* and the rights object into one *DRM* Message. The DRM Message is made available to the device (e.g. by publishing it on a Web page).
2. HTTP Download or OMA Download is used to download the DRM Message (*media object* + *rights object*).
3. The User is using the media object. The DRM agent ensures that the *media object* is used according to the rights.

Extensions:

None.

6.3.3 Use Case DRM-3: Separate Delivery (Push)

A protected *media object* is downloaded from an HTTP server. Later, the corresponding *rights object* is delivered to the device via WAP Push (e.g. over SMS).

When this case ends, if the main scenario is completed, the *media object* is available in the device. It can be used only according to the granted usage rights.

Main scenario:

1. The Content Provider, using a *DRM Packager*, packages the encrypted *media object* into a *DRM Content format*. The key to decrypt the *media object* is put into the *rights object.* The *DRM Content format* URL is made available to the device (e.g. by publishing it on a Web page). The *rights object* is not available to the device.
2. HTTP Download or OMA Download is used to deliver the *DRM Content format*.
3. The User waits for the *rights object* to be delivered via a WAP Push message from the Content Provider.

Note: If OMA Download was used, the status report could be used as a "trigger" in the network to push the *rights object* to the device, after a successful download of the encrypted *media object*.

4. The *media object* is decrypted using the key from the *rights*.

5. The User is using the media object. The DRM agent ensures that the *media object* is used according to the rights.

Extensions:

None.

Appendix A. HTTP and MIDP Download (Informative)

OMA Download is based on HTTP and MIDP Download. These technologies are explained in this section.

A.1 Basic HTTP Download

Today low-value media objects are downloaded with the standard Web protocol, HTTP [RFC2616], or the binary WAP variant, WSP [WAP230]. This basic download service transfers media objects from the Web to the device in one single network transaction, a GET request and response.

Basic Download is, because nothing more than standard HTTP or WSP is used, certain to be supported by all Web clients, including all WAP phones. It is exactly the same technology that is used to download Web pages. For many cases of downloading, Basic Download is sufficient.

A.1.1 Using HTTP for Download

Specifically for downloading, the following standard HTTP features are worth pointing out:

- The size of the media object is available to the device in the beginning of the GET response. If the size is larger than what the device can handle, the device can abort the GET response by simply closing the network connection (letting the server time-out).
- If the device knows that a URL is a media object to be downloaded, it can send a HEAD request before it sends the GET request. The HEAD response returns the HTTP header information for the media object at the URL, including the size and media type. The header information can be used to determine whether the device will be able to handle the media object, and ask the user for a confirmation before a GET request is sent to the URL. It would be inefficient if the device always used the HEAD request before every GET request. So this is useful only under certain circumstances; for example, an application on the device dedicated to download large images or if the device is able to use some heuristics[2] to determine the purpose of a URL before it is requested.
- The server can use the *Content-Disposition* response header (see [RFC2616] section 19.5.1), to suggest a default filename, if the media object is saved on a filesystem. (See [RFC2616] section 15.5 for security issues.)

In addition to the features directly related to the transfer of media objects, HTTP enables the following commonly used services:

- **Discovery** using Web pages; authored in the XHTML Mobile profile WAP298], for example.
- **Authentication** using HTTP authentication [RFC2617].
- **State management** using HTTP "cookies" [WAP223][RFC2965]. To maintain state information (e.g. a session identifier) between the different "steps" in the downloading procedure. An alternative, more reliable because users may disable cookies, is to put the state information into the query-part of the URL.
- **Transport level security** using TLS [RFC2246]. To authenticate the client or the server, and to encrypt media objects during transport.

---

[2] One way by which the device can know that a URL is something that the user would like to download and save on the device is by looking at the "type" attribute on the XHTML anchor element [WAP298].

```
<a href="anObject.ext" type="image/gif">Download image</a>
```

Media types such as "image/gif" and "application/vnd.sun.j2me.app-descriptor" indicates to the device that the target of the link is not just another Web page, but a specific media object that the user potentially want to download and store on the device. The file extension of the URL could also be used to "guess" the media type, however, much less reliable than an attribute specifically intended to specify the media type.

Basic HTTP Download is the basis for MIDlet Download and OMA Download. These services extend Basic Download with more functionality before and after media object transfer. So Basic Download is a significant part of the MIDlet Download and OMA Download services.

A.1.2 When is HTTP Insufficient?

The functionality provided by Basic HTTP Download is insufficient in the following situations:

- When it is important that the device makes capability checks (e.g. memory, media type) to make sure it supports the media object and that the user is given an opportunity to confirm the download based on the results of the capability checks, before the media object is requested from the server;
- When it is important that the content provider gets an indication of the outcome of the downloading, after the media object has been requested from the server.

Standard HTTP is impractical, in some cases even impossible, to use in these situations. Here is why.

Before the media object is transferred from the server, the client does not know its size or media type, thus cannot prevent the downloading of media objects that are too large; expect by abruptly closing the connection while the object is in transfer and letting the server time-out. Before getting the GET response from the server, the device typically knows nothing more about the media object than the URL. To a limited extent, HTTP content negotiation (see [RFC2616]) and UAProf [WAP248] can be used to let the server choose an appropriate variant of the media object. But the server can have only limited knowledge of the device's capabilities. The device may have megabytes of free memory, but may be incapable of supporting GIF images larger than 3KB. If the media object takes 80-90% of all free memory, the user may want to download something different (smaller). With HTTP content negotiation, the server makes most of the decisions – it is server driven. The need for the user and the device to make decisions before the actual downloading starts, make HTTP content negotiation insufficient. Another limitation is the lack of confirmation back to the server, after the download. The HTTP server cannot tell for sure the outcome of the download. Did the device get the complete media object? Did the user abort the download? Without this knowledge, useful as the basis for billing and quality assurance, the content provider and the operator has no simple way of finding out which media objects have been successfully downloaded and which have not.

The HEAD method can be used by the device to request from the server metadata (actually HTTP headers) such as media type and size about the media object; the same kind of information that is available in the *download descriptor*. It is not practical, however, to always send an HEAD before a GET request is sent to the server. The problem is that the device cannot know when a URL is a downloadable media object (and it is appropriate to use the HEAD method), and when it is a normal Web page (and not appropriate to use the HEAD method).

A.2 Java™ MIDlet Download

The downloading of Java™ MIDlets has demonstrated the need for extensions of Basic Download. At least for some media types, more steps than just media object transfer are needed. The introduction of higher-value and larger-sized media objects underscores the need for an enhanced downloading service. The popularity of ring-tones, games, themes, and images, puts more requirements on the overall user experience of upgrading the phone with new media objects, and the ways by which the operator and content provider can monitor and charge for it.

The Java™ MIDP [MIDP] specification defines one way of downloading MIDlets that is possible to use as the basis for the downloading of other content types. It extends Basic Download by adding two additional steps, one before the media object transfer and one after. The additional steps are:

- **Before downloading** of the MIDlet, a description of the MIDlet is downloaded. The description is a small file, the Java™ Application Descriptor (JAD), containing metadata about the MIDlet and instructions to the client, the Java™ Application Manager (JAM), for how to download the MIDlet.

  The MIDlet is downloaded using standard Basic HTTP Download.

- **After downloading** the MIDlet, a status report is posted to a Web site, indicating the outcome of the download.

With these simple extensions to Basic Download, MIDlet Download can provide a good downloading experience to the user and control of the downloading procedure to the content provider.

Because MIDlet Download contains MIDP-specific elements such as MIDP-version, it cannot be fully replaced with a general download framework, such as OMA Download, that does not have those elements. Since already deployed MIDlet Download clients will require the continuation of MIDlet Download servers, the intention is not to use OMA Download as the primary mechanism for MIDlet Download.

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
OMA-Download-ARCH-v1_0-20020610-a	10 Jun 2002	The initial version of this document.
OMA-Download-ARCH-V1_0-20040625-A	25 Jun 2004	Status changed to Approved by TP Ref TP Doc# OMA-TP-2004-0155-DL-V1_0-for-final-approval

B.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
n/a	n/a	n/a	No previous version in OMA