Implementation Best Practices for OMA DRM v1.0 protected MIDlets

Version 1.0
May 5th , 2004

Change History:

Version	Date	Author	Comments:
1.0	05.05.2004	Companies who have undersigned	First public release

NOTICE

DISCLAIMER OF WARRANTIES

TABLE OF CONTENTS

GLOSSARY AND DEFINITIONS

Term	Definition
AMS	Application Management Software as described in the MIDP specification [MIDP20]
DCF	DRM Content Format [DCF] used in the OMA DRM separate delivery method. The format for encrypted media object and associated meta data. Rights object containing information about consumption rules is not stored in the DCF file
DRM message	A message as defined in [DRM]: a message containing a media object and an optional rights object. Media objects received inside a DRM message must not leave the device. The optional rights object defines additional consumption rules for the media object
HTTP	HyperText Transfer Protocol
JAD	Java Application Descriptor, a text file containing meta-, signing and provisioning information about the MIDlet JAR file. Defined in the MIDP specification [MIDP20]. A JAD file contains the MIDlet-Jar-URL attribute that contains a reference to the actual file containing the Java application. See MIDlet suite
JAR	Java ARchive file. A possibly data compressed file format containing the required class and resource (e.g. images, sounds) files for an application(s). It has “jar” file extension. Present document also defines how JAR file format is used to store JAD and JAR files (see MIDlet Message)
Manifest	A text file encoded with UTF-8 inside a JAR file. Defines meta-information about the JAR content. If a JAR file contains MIDlet suite class files then the manifest file contains many same attributes as the JAD file. See the MIDP specification [MIDP20] for actual attributes stored in the Manifest
MIDlet Message	A JAR file received via some connectivity technology that contains a MIDlet suite (a JAD file and a JAR (possibly in the DCF) file). The actual format of the MIDlet Message is defined in Section 5.3 of this specification
MIDlet suite	A collection of MIDP applications bundled together as a single unit. Depending on a context the *MIDlet suite* refers either only to the JAR file or both to JAD and JAR files
MIME type	A media type used in various Internet protocols like MIME and HTTP. E.g. “text/plain” MIME type indicates plain text content. Often the MIME type is carried in the Content-type header. MIME types are registered in IANA registry for Internet media types. MIME type is also known as Internet media type
OMA	Open Mobile Alliance
OMA DRM	OMA Digital Rights Management
OPP	The Bluetooth Object Push Profile – a profile intended for sending (and sometimes) receiving of objects via the Bluetooth wireless technology. See [OPP]
OTA	Over The Air
Rights	A term used in OMA DRM specification [DRM] to describe permissions and constraints that define under which circumstances access is granted to a media object
RMSD	Removable Mass Storage Device
Superdistribution	A term used in OMA DRM specification [DRM] for content distribution model that allows devices to forward the encrypted content and the receiving end to obtain rights for that content
UI	User Interface
URL	Uniform Resource Locator

Table 1. Glossary

REFERENCES

[DCF] DRM Content Format Version 1.0, OMA-Download-DRMCF-v1_0-20031113-C, Open Mobile Alliance. Available at: http://www.openmobilealliance.org/

[DLOTA] Generic Content Download Over The Air Specification, Version 1.0, OMA-Download-OTA-v1_0-20030221-C, Available at: http://www.openmobilealliance.org/

[DRM] Digital Rights Management Version 1.0, OMA-Download-DRM-v1_0-20031031-C, Open Mobile Alliance. Available at: http://www.openmobilealliance.org/

[DTV] "DTV Application Software Environment Level 1 (DASE-1). Part 5: ZIP Archive Resource Format. ATSC Standard", Advanced Television Systems Committee, 09 March 2003. Available at: http://www.atsc.org/standards/a_100_5.pdf

[GZIP] RFC 1952, GZIP file format specification v4.3. Available at: http://www.ietf.org/rfc/rfc1952.txt

[IBP] Implementation Best Practices For OMA DRM release 1, V1.1. August 2003, Nokia, Motorola, Siemens, SonyEricsson, Vodafone, T-Mobile.

[JAR] "JAR File Specification. J2SE 1.3 documentation", Sun Microsystems. Available at: http://java.sun.com/j2se/1.3/docs/guide/jar/jar.html

[JSR120] Wireless Messaging API, Java Community Process. Available at: http://www.jcp.org/en/jsr/all

[JSR205] Wireless Messaging API 2.0, Java Community Process. Available at : http://www.jcp.org/en/jsr/all

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

[OPP] "Specification of the Bluetooth System", Specification Volume 1. Profiles, Part K11. Version 1.1.

[REL] Rights Expression Language Version 1.0, OMA-Download-DRMREL-v1_0-20031031-C, Open Mobile Alliance. Available at: http://www.openmobilealliance.org/

[ZIP] Info-ZIP Application Note 970311. Available at: ftp://ftp.uu.net/pub/archiving/zip/doc/appnote-970311-iz.zip

1. INTRODUCTION

This document addresses usage of OMA DRM technology for protection of Java MIDP MIDlet suites. The aim of the document is to increase the interoperability of implementations adhering to the OMA DRM release 1 specifications by clarifying parts of the specifications where there is an opportunity for divergent implementations.

In particular, the document describes the following:

- Usage of MIDP OTA download for delivery of MIDlet suites protected using various OMA DRM methods
- OMA DRM consumption permissions applicable to MIDlet suites and their enforcement
- Access to OMA DRM protected content from MIDlets
- Local OMA DRM superdistribution of MIDlet suites
- Device capability checking by the server during rights acquisition procedure

The present document is a technical specification indicating how the client and the provisioning server must implement the features listed above. The exact set of features to be supported by the individual devices shall be specified by interested parties in complementing documentation, i.e. relevant (operator) mobile terminal requirement documents.

2. OTA PROVISIONING OF OMA DRM PROTECTED MIDLET SUITES

This chapter describes how MIDP OTA download is used to deliver MIDlet suites protected by OMA DRM 1.0 forward-lock, combined delivery and separate delivery methods.

The chapter does not cover OMA DRM superdistribution and does not describe how rights for superdistributed suites are obtained. For more information on superdistribution of OMA DRM protected MIDlet suites, see Chapter 5.

2.1 OMA DRM related attributes in the JAD file

OTA provisioning of OMA DRM protected MIDlet suites in some cases requires addition of new attributes to the JAD file. Table 2 lists and describes attributes that can be added:

JAD file attribute name	Description	Server support	Device support
[Oma-Drm-Delivery-Notify] 2.1.r1	Contains a URL to which the device reports the outcome of download of OMA DRM protected JAR file to trigger sending of the rights object	Optional	Mandatory
[Oma-Drm-Package-Size] 2.1.r2	Contains the size of OMA DRM package (DRM message or DCF file) containing a JAR file	Mandatory	Optional

Table 2. OMA DRM related attributes in the JAD file

2.2 Forward-lock

This section describes how MIDP OTA download is used to deliver MIDlet suites protected by the OMA DRM forward-lock method.

[OMA DRM forward-lock protected MIDlet suite has its JAR file wrapped into the DRM message]2.2.r1, therefore a URL from the [MIDlet-Jar-URL attribute of the JAD file points to the DRM message]2.2.r2. From the user points of view the download process looks the same as with a non-protected MIDlet suite.

[In case when the JAD file is present, it is downloaded first (download is initiated from some discovery application, e.g. a browser). Then, if the user approves download of the MIDlet suite, the device uses a URL from the MIDlet-Jar-URL attribute of the JAD file to request the DRM message]2.2.r3. [If there is no JAD file available, the discovery application (browser) immediately initiates download of the DRM message]2.2.r4. In any case, the provisioning server answers with a DRM message (application/vnd.oma.drm.message MIME type) containing a plaintext JAR file, but no rights object. The device downloads the DRM message.

[After the DRM message is in the device, the process continues as in MIDP OTA download]2.2.r5, i.e. all necessary verifications are made. Note, [that the value of the MIDlet-Jar-Size attribute of the JAD file is always compared with the size of the JAR file]2.2.r6, not with the size of the DRM message.

[MIDP installation status reporting is also done as normal, standard MIDP status codes are used]2.2.r7. However, there is a need for one new code – [for the case when the received DRM message is invalid (i.e. the device cannot parse it). In such situation, code 953 – “Non-Acceptable Content” [DLOTA] is used]2.2.r8.

Figure 1 shows how OMA DRM forward-lock protected MIDlet suites are delivered using MIDP OTA download.

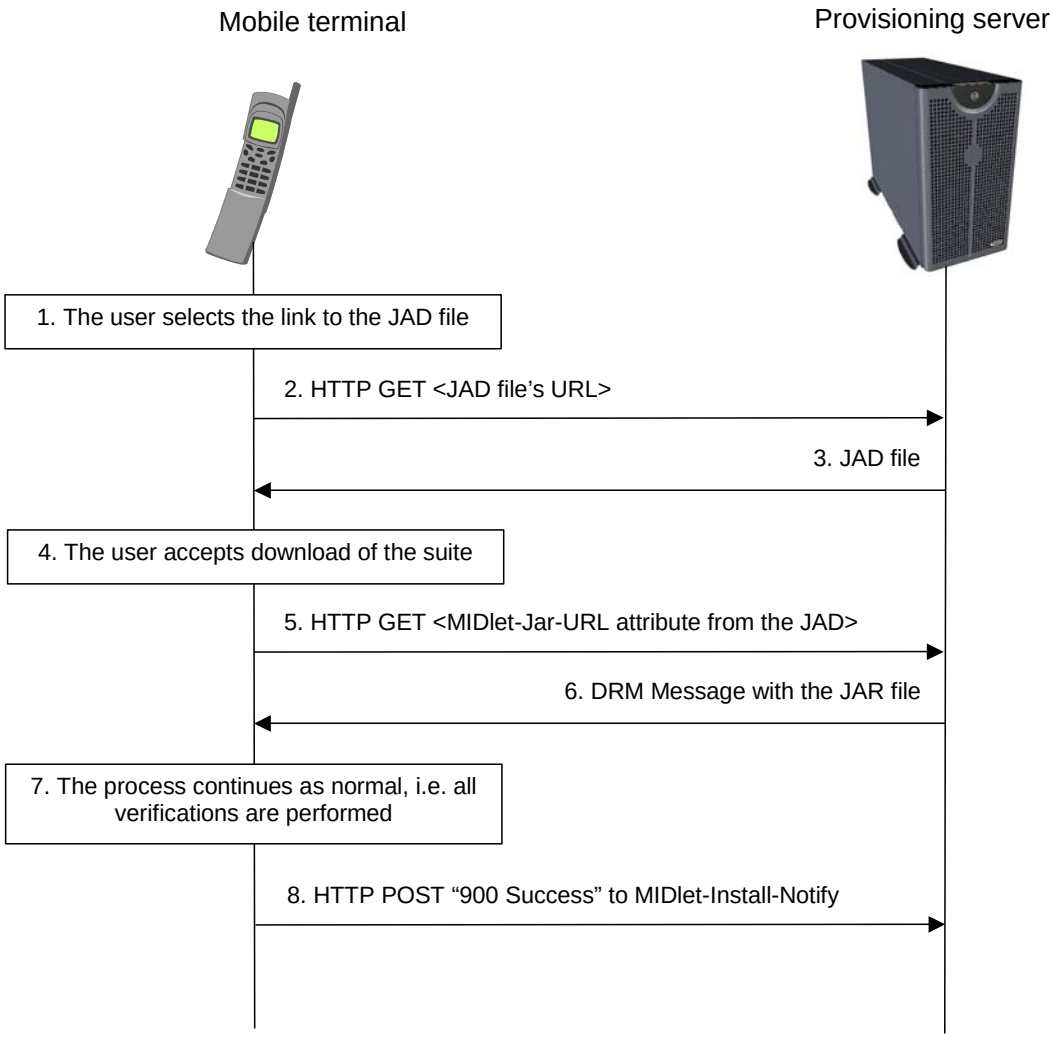

Figure 1. MIDP OTA download of the OMA DRM forward-lock MIDlet suite

Note, that according to the OMA DRM specification [DRM], a DRM message that does not contain a rights object can contain a DCF file. This case is described in Section 2.4.1 of this document.

2.3 Combined delivery

This section describes how MIDP OTA download is used to deliver MIDlet suites protected by the OMA DRM combined delivery method.

[OMA DRM combined delivery protected MIDlet suite has its JAR file wrapped into the DRM message (and accompanied with the rights object)]$_{2.3.r1}$, [therefore a URL from the MIDlet-Jar-URL attribute of the JAD file points to the DRM message]$_{2.3.r2}$. From the user points of view the download process looks the same as with a non-protected MIDlet suite.

[If the JAD file is present, it is downloaded first (download is initiated from some discovery application, e.g. a browser)]2.3.r3. [Then, if the user approves download of the MIDlet suite, the device uses a URL from the MIDlet-Jar-URL attribute of the JAD file to request the DRM message]2.3.r4. [If there is no JAD file available, the discovery application (browser) immediately initiates download of the DRM message]2.3.r5. [In any case, the provisioning server answers with a DRM message (application/vnd.oma.drm.message MIME type) containing a rights object and a plaintext JAR file. The device downloads the DRM message]2.3.r6.

[After the DRM message is in the device, the rights object is stored. Then the installation process continues as normal, i.e. all necessary verifications are made]2.3.r7. [Note, that the value of the MIDlet-Jar-Size attribute of the JAD file is always compared with the size of the JAR file, not with the size of the DRM message]2.3.r8.

[MIDP installation status reporting is also done as normal, standard MIDP status codes are used]2.3.r9. Similar to the forward-lock method, [code 953 – “Non-Acceptable Content” [DLOTA] is used to report that the downloaded DRM message is invalid (i.e. the device cannot parse it)]2.3.r10.

Figure 2 shows how OMA DRM combined delivery protected MIDlet suites are delivered using MIDP OTA download.

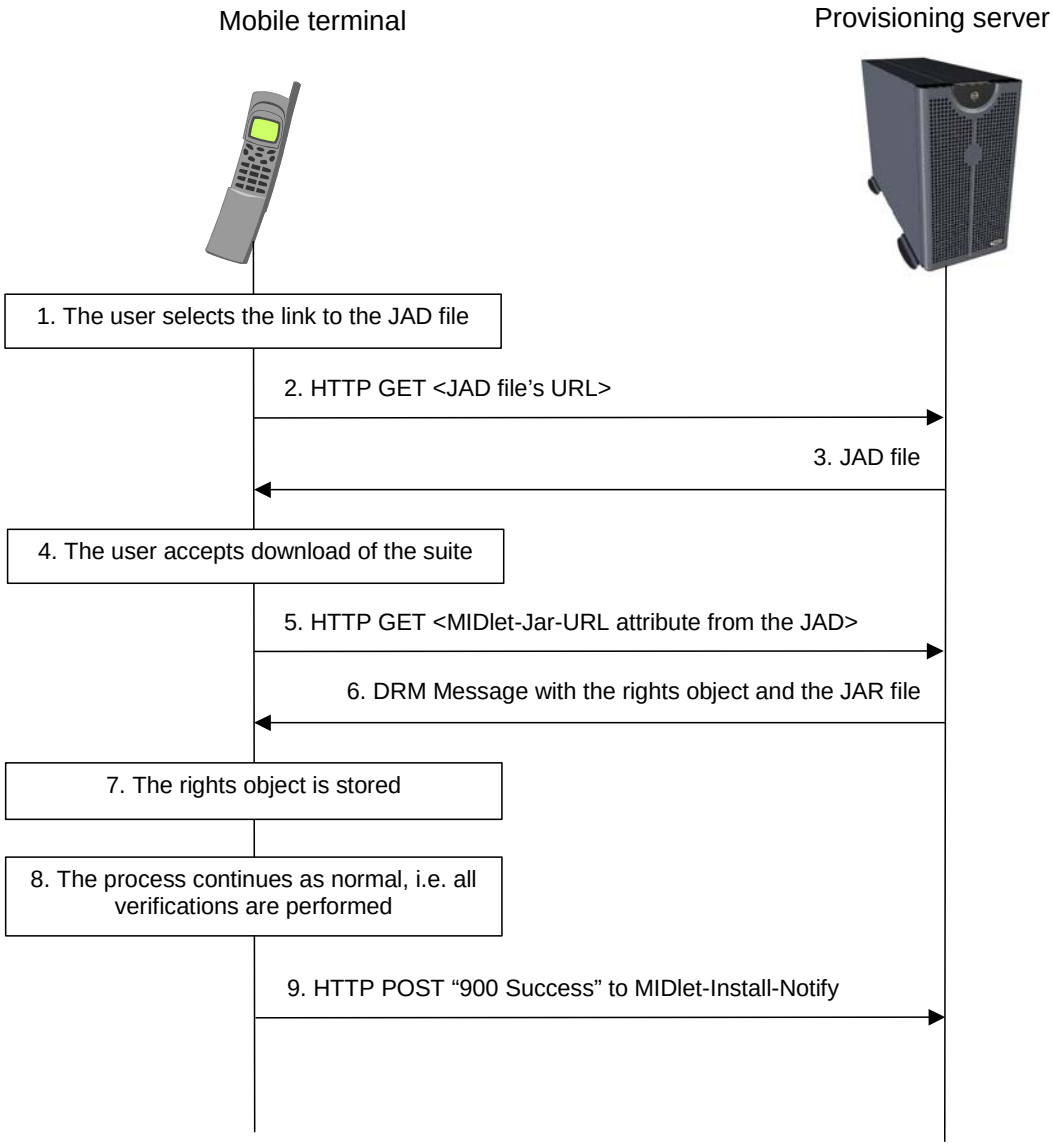

Figure 2. MIDP OTA download of the OMA DRM combined delivery MIDlet suite

Note, that according to the OMA DRM specification [DRM], a DRM message that contains a rights object cannot contain a DCF file.

2.4 Separate delivery

This section describes how MIDP OTA download is used to deliver MIDlet suites protected by the OMA DRM separate delivery method.

[OMA DRM separate delivery protected MIDlet suite has its JAR file in the DRM Content Format [DCF]]2.4.r1, therefore a [URL from the MIDlet-Jar-URL attribute of the JAD file points to the DCF file]2.4.r2.

[If the JAD file is present, it is downloaded first]2.4.r3 (download is initiated from some discovery application, e.g. a browser). [Then, if the user approves download of the MIDlet suite, the device uses a URL from the MIDlet-Jar-URL attribute of the JAD file to request

the DCF file]2.4.r4. [If there is no JAD file available, the discovery application (browser) immediately initiates download of the DCF file]2.4.r5. [In any case, the provisioning server answers with a DCF file (application/vnd.oma.drm.content MIME type) containing a JAR file]2.4.r6. [The device downloads the DCF file. When sending a DCF file the provisioning server can include the X-Oma-Drm-Separate-Delivery header]2.4.r7. [If the header is present, the device saves the value of the header for later use]2.4.r8.

[If the JAD file contains the Oma-Drm-Delivery-Notify attribute with a valid URL, the device posts a report about the outcome of DCF file download to the server]2.4.r9. [Reporting is done using HTTP POST to the URL from the Oma-Drm-Delivery-Notify attribute with the following status codes]2.4.r10:

Status Code	**Status Message**
900	Success
901	Insufficient memory
902	User cancelled
903	Loss of Service
953	Non-Acceptable Content

Table 3. Status codes for download reporting

[In response to the download report the server replies with a “200 OK” response. No content is sent in the server’s response, but if sent it is ignored by the device]2.4.r11. [If any response was received from the server, sending is not retried. If the download report cannot be sent (e.g. due to loss of network connectivity) or if the server reply is not received, sending should be retried and provisioning of the suite continues as normal]2.4.r12.

[If there was an error while downloading a DCF file, or if the downloaded DCF file is invalid, the user is informed. Otherwise the process continues described below]2.4.r13.

After the report (900 Success) was sent and [if the X-Oma-Drm-Separate-Delivery header was present in the provisioning server’s response with the DCF file (see above), the device starts waiting for the rights object to be pushed]2.4.r14. [In the situation when the X-Oma-Drm-Separate-Delivery header is missing from the server’s response, no rights object will be pushed to the device]2.4.r15. This means that the user needs to acquire rights separately (which is beyond the scope of this document).

[Upon the receiving of the status report of the OMA-Drm-Separate-Delivery notify, the provisioning server may issue the rights object by pushing it to the device]2.4.r16.

[When waiting for the rights object to be pushed, the device uses the following algorithm to determine the waiting time]2.4.r17:

```
#define MAX_DEVICE_TIME_OUT  <device_specific_value>


If (0 < X-Oma-Drm-Separate-Delivery header value < MAX_DEVICE_TIME_OUT )
                wait_time = X-Oma-Drm-Separate-Delivery header value
else
                wait_time = MAX_DEVICE_TIME_OUT
```

MAX_DEVICE_TIME_OUT can be a device specific value and it is outside the scope of this document to define it. Also, it is up to the provisioning server to decide which value to put into X-Oma-Drm-Separate-Delivery header.

Rights object is received in time

[If the rights object is received by the device before waiting time elapses, the process continues as normal MIDP OTA download]2.4.r18, i.e. all necessary verifications are made. Note, that the value of the [MIDlet-Jar-Size attribute of the JAD file is always compared with the size of the JAR file, not with the size of the DCF file]2.4.r19.

[MIDP installation status reporting is also done as normal (i.e. to the MIDlet-Install-Notify URL from the JAD file), standard MIDP status codes are used]2.4.r20. [Additional error code is needed to report situations where the received rights object for the given DCF file contains errors and thus the DCF file cannot be open. E.g., the rights object contains a wrong decryption key. Code 953 – “Non-Acceptable Content” is used [DLOTA] to report such situations]2.4.r21.

In case of timely arrival of the rights object, the whole download process is perceived by the user as usual MIDP OTA download.

[In situations when there are multiple rights objects for the same OMA DRM separate delivery protected MIDlet suite the behaviour is the same as defined in the Implementation Best Practices for DRM 1 document [**Error! Reference source not found.**]]2.4.r22.

Rights object does not arrive in time

[If the rights object was not received during the waiting interval, the user is informed that rights did not arrive]2.4.r23. [The user then has the choice either to start the rights acquisition procedure (which is beyond the scope of this document) immediately or to do it later]2.4.r24. [If the user selects to acquire rights later, MIDlet suite files (DCF and JAD, if present) are saved in the device]2.4.r25. [Such MIDlet suites (downloaded, but not installed due to the absence of the rights object) are accessible from the device UI, so the user can always start the rights acquisition procedure or delete the suite]2.4.r26.

[It can happen so, that for some MIDlet suite the rights object will be pushed late, i.e. beyond the waiting period. In this situation the installation is continued (with or without user interaction) from where it stopped (after DCF file download), i.e. all MIDP checks are done, MIDP installation status report is sent, etc]2.4.r27.

Figure 3 shows how OMA DRM separate delivery protected MIDlet suites are delivered using MIDP OTA download. Note, that step 7 (reporting of DRM message download) happens only if the Oma-Drm-Delivery-Notify attribute is present in the JAD file.

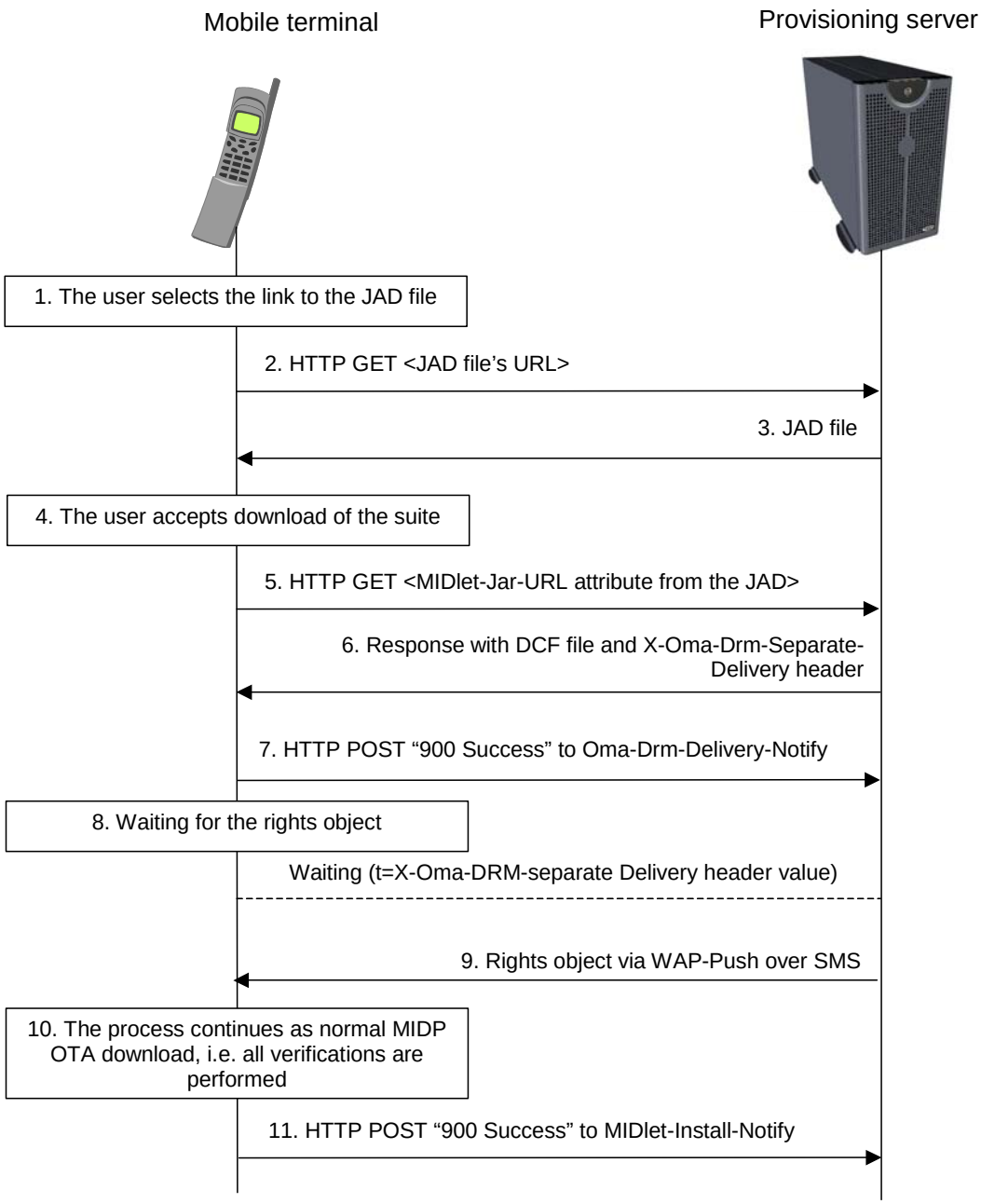

Figure 3. MIDP OTA download of the OMA DRM separate delivery MIDlet suite

As defined by the OMA DRM specification [DRM], MIDlet suites protected by the OMA DRM separate delivery method can leave the device (unless they have been sent inside a DRM message, see Section 2.4.1). The exact mechanism for OMA DRM superdistribution of such suites can be found in Chapter 3 of the present document.

2.4.1 Special case: forward-locked JAR file in the DCF

The OMA DRM specification [DRM] contains provisions for delivery of DCF files inside DRM messages (“forward-locked DCF” case). This section describes how OTA delivery happens in this case.

In this download scenario, [the JAR file is in the DCF, and the DCF file, in turn, is inside the DRM message without the rights object]2.4.1.r1. [The rights object for the DCF is pushed separately]2.4.1.r2. In essence, the download process goes as in case with usual OMA DRM separate delivery protected suites (Section 2.4), the only difference is that a DRM message is downloaded instead of a DCF file. [Note, that download notification (to a URL from the Oma-Drm-Delivery-Notify attribute of the JAD file) is done in the same way as described in Section 2.4]2.4.1.r3.

[Superdistribution does not apply to MIDlet suites delivered using this method]2.4.1.r4.

Figure 4 shows how MIDP OTA download is used to deliver MIDlet suites with the JAR file inside the forward-locked DCF. Note, that step 7 (reporting of DRM message download) happens only if the Oma-Drm-Delivery-Notify attribute is present in the JAD file.

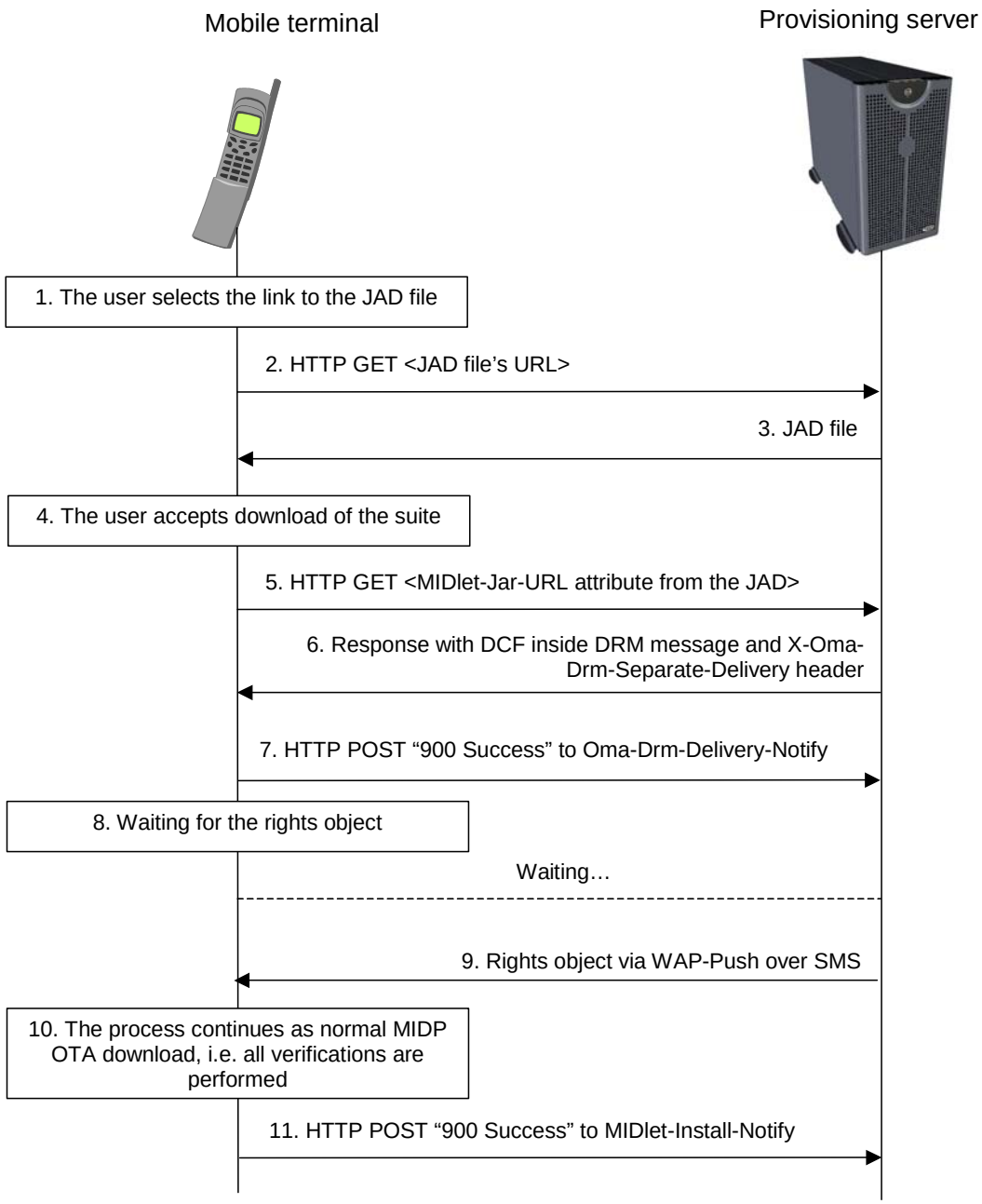

Figure 4. MIDP OTA download of the MIDlet suite with the JAR file inside the forward-locked DCF

2.5 Update of MIDlet suites

Update of OMA DRM protected and non-OMA DRM protected MIDlet suites happens as defined in the MIDP specification [MIDP20].

2.6 Suites with invalid and non-existent rights

[The device UI has to allow the user to view and delete MIDlet suites without rights or with invalid rights]2.6.r1. [For separate delivery protected suites the device UI has to allow the

user to request rights for the suite][2.6.r2]. [It is up to the user delete or preserve superdistributed MIDlet suites without rights][2.6.r3].

3. PERMISSIONS FOR MIDLET SUITES

Among four permissions (<play>, <display>, <execute>, <print>) defined in the Rights Expression Language specification [REL], only <execute> applies to MIDlet suites. This means that [<execute> is the only permission the device takes into consideration when making decision about granting access to the MIDlet suite, all other permissions are ignored][3.r1].

[From three constraints (<count>, <datetime>, <interval>) defined in the same specification, all three can be used in conjunction with the <execute> permission][3.r2].

[Permissions for MIDlet suites delivered via forward-lock method is always <execute>, without any constraints][3.r3]. [Permissions for suites delivered via combined delivery or separate delivery methods are read from corresponding rights objects][3.r4].

[Value in the <count> constraint denotes how many times the MIDlet suite as a whole can be accessed by the user (according to the OMA DRM specification [DRM])][3.r5]. E.g., if some suite contains several MIDlets, and there is the <execute> permission with the <count> constraint equal to 1, this means that only one MIDlet from the suite can be executed and only once.

There is a need to define when and how consumption permissions for MIDlet suites are enforced. Here are the rules:

- [Permissions and constraints are checked each time MIDlets from the suite are launched. If rights are invalid, MIDlets from the suite are not launched][3.r6].

4. ACCESS TO OMA DRM PROTECTED CONTENT FROM MIDLETS

[Only MIDlets from MIDlet suites belonging to operator and manufacturer domains can access OMA DRM protected content (images, music files, etc)][4.r1]. In here, access to a particular piece of OMA DRM-protected content is regulated by consumption rules for this content.

[MIDlets that don't belong to manufacturer or operator domains cannot access OMA DRM protected content in plaintext][4.r2].

[Access to a particular piece of OMA DRM protected content from a MIDlet happens according to consumption rules for this content][4.r3]. [MIDlets cannot access rights objects][4.r4]. E.g., if Wireless Messaging API 1.0 or 2.0 ([JSR120], [JSR205]) is present on the device, [MIDlets cannot receive or access WAP Push SMS messages that contain rights objects][4.r5].

5. LOCAL SUPERDISTRIBUTION OF OMA DRM PROTECTED MIDLET SUITES

This chapter is devoted to local superdistribution of OMA DRM protected MIDlet suites. In particular it describes mechanisms and formats for sending and receiving of OMA DRM protected suites. Device-to-device superdistribution is the only case considered.

[OMA superdistribution is allowed only for the content that was delivered using OMA separate delivery method unless the DCF containing the JAR file was received in a DRM message without rights (i.e. it is forward-locked)]5.r1.

5.1 Local sending of MIDlet suites

This section describes how MIDlet suites are sent from one device to another (OMA superdistribution) via a local connectivity technology.

5.1.1 Sending process

[If the device supports OMA superdistribution via local connectivity technologies (Bluetooth, Infrared, etc), then it is possible to forward at least installed MIDlet suites that are protected by the OMA DRM separate delivery method]5.1.1.r1. [The device can also support forwarding of not installed suites protected by the OMA DRM separate delivery method]5.1.1.r2.

Forwarding of eligible suites happens in the following manner:

1. [Presence of the JAD file is checked (decision on availability of the JAD file for a given MIDlet suite is made according to Section 5.1.2)]5.1.1.r3.

   [If the JAD file is present and contains the JAR file's digital signature and the device contains the rights object to access the JAR in the DCF, the integrity of the JAR file is checked before sending the suite. Normal MIDlet suite authentication procedure described in Chapter 4 of the MIDP 2.0 specification [MIDP20] is used. If the integrity check fails, the suite is not sent. An error message is shown to the user]5.1.1.r4.

   [If the JAD file is missing, only the DCF file with the JAR file inside is sent from the device and item 2 below is ignored]5.1.1.r5.

2. [JAD and DCF files are wrapped in the MIDlet Message format according to rules described in Section 5.3 of this specification]5.1.1.r6. More precisely:

   - [JAD and DCF are placed to the container JAR file. Plain structure is used, i.e. the JAD and DCF are placed to in the root of the container JAR file (no folders)

   - Three manifest attributes defined in Section 5.3 are placed to the manifest of the container JAR file. Distribution-Package attribute contains "1.0" string. Distribution-Descriptor-Filename attribute contains the name of the JAD file. Distribution-Content-Filename attribute contains the name of the DCF file

   - The name of container JAR file is the value of MIDlet-Name attribute in the JAD file. The extension is "jar"]5.1.1.r7.

3. [An attempt is made to send the DCF file or the MIDlet Message via the selected bearer. Rules for sending of DCF files and MIDlet Messages via OBEX are defined in Section 5.4.1]5.1.1.r8.

Note, that a superdistributed MIDlet suite may not work on the other device due to multiple reasons, e.g.: different screen resolution, different keypad, different Java APIs, etc. [Therefore, the user of the sending device has to be notified about this risk. This is done by displaying an appropriate prompt before sending of a suite]5.1.1.r9.

5.1.2 JAD file availability

As devices vary greatly, the end-user can have different opportunities to select and forward a MIDlet suite (e.g. from inside of AMS menus, from the folder for incoming messages, from the file browser, from inside of an MMS or an e-mail message to which suite's files are attached, etc). Because of that a decision whether a given MIDlet suite has a JAD file is made case by case. [Each time the MIDlet suite is superdistributed every attempt is made to locate the JAD file and send it]5.1.2.r1.

5.1.3 Bluetooth wireless technology

When the MIDlet suite is sent (either as a MIDlet Messages or as a DCF file) via the Bluetooth Wireless Technology, the Object Push Profile [OPP] is used. The procedure is as follows:

1. [The user selects the target device among those found using Bluetooth Device Discovery

2. The Object Push Profile is searched on the selected device (via Bluetooth Service Discovery). If the profile is not present, sending fails and an appropriate error message is shown to the user. If the profile is present, its service record attributes are fetched and analyzed

3. Supported Formats List attribute of the profile's service record is analyzed. Sending is possible only when the receiving device indicates support for MIDlet Messages (or DCF files). E.g., when the above-mentioned list includes the value 0xFF (any type of object). If receiving of MIDlet Messages is not supported, the MIDlet suite is not sent. An error message saying that the other party does not support receiving of MIDlet suites is shown to the user of the sending device.

   If sending of MIDlet Messages and DCF files is possible, it happens as described in Section 5.4.1]5.1.3.r1.

5.1.4 Infrared and other local connectivity technologies

[If the receiving device refuses to accept a MIDlet suite (in the form of a MIDlet Messages or in the form of a DCF file), sending fails]5.1.4.r1. [An error message saying that the other party does not support receiving of MIDlet suites is shown to the user of the sending device]5.1.4.r2.

If OBEX is used as a sending protocol, sending of MIDlet Messages and DCF files happens as described in Section 5.4.1.

5.2 Local receiving of MIDlet suites

This chapter describes receiving of superdistributed MIDlet suites via local connectivity technologies.

5.2.1 Supported formats

The following formats are supported:

- [MIDlet Message (described in Section 5.3), with the only JAR file in the DCF. Receiving of MIDlet Messages via OBEX is described in Section 5.4.2]5.2.1.r1
- [DCF file with the only JAR file inside. Receiving of DCF files via OBEX is described in Section 5.4.2]5.2.1.r2.

5.2.2 Bluetooth wireless technology

[When receiving MIDlet suites via the Bluetooth Wireless Technology the Object Push Profile [OPP] is used]5.2.2.r1. The device supports receiving of MIDlet suites according to Section 5.2.1 of this document. Receiving of MIDlet Messages and DCF files via OBEX is described in Section 5.4.2.

[There is a need to indicate the support for receiving of MIDlet suites in a particular format via the Object Push Profile service record. Thereto the value 0xFF is added to the Supported Formats List attribute of the OPP service record (if it is not yet there)]5.2.2.r2. This will indicate that the device can receive any types of files (including those listed in Section 5.2.1).

5.2.3 Infrared and other local connectivity technologies

[When receiving MIDlet suites via Infrared or other local connectivity technologies the device supports MIDlet suite formats according to Section 5.2.1 of this document. Receiving of MIDlet Messages and DCF files via OBEX is described in Section 5.4.2]5.2.3.r1.

5.2.4 Installation

The user has a possibility to install a MIDlet suite contained in the received DCF file or MIDlet message.

[In here, suite installation is understood as a process resulting in ability of the user to run MIDlets from the suite]5.2.4.r1. [This process includes MIDP verifications, possible acquisition of rights, etc. MIDP installation status reporting and deletion notification are not done for superdistributed suites]5.2.4.r2.

5.3 MIDlet Message

This section defines a transfer format for MIDlet suites - the MIDlet Message. It is a JAR file that contains a JAD file and a JAR file in the DCF.

Figure 5 presents a logical view of a MIDlet Message.

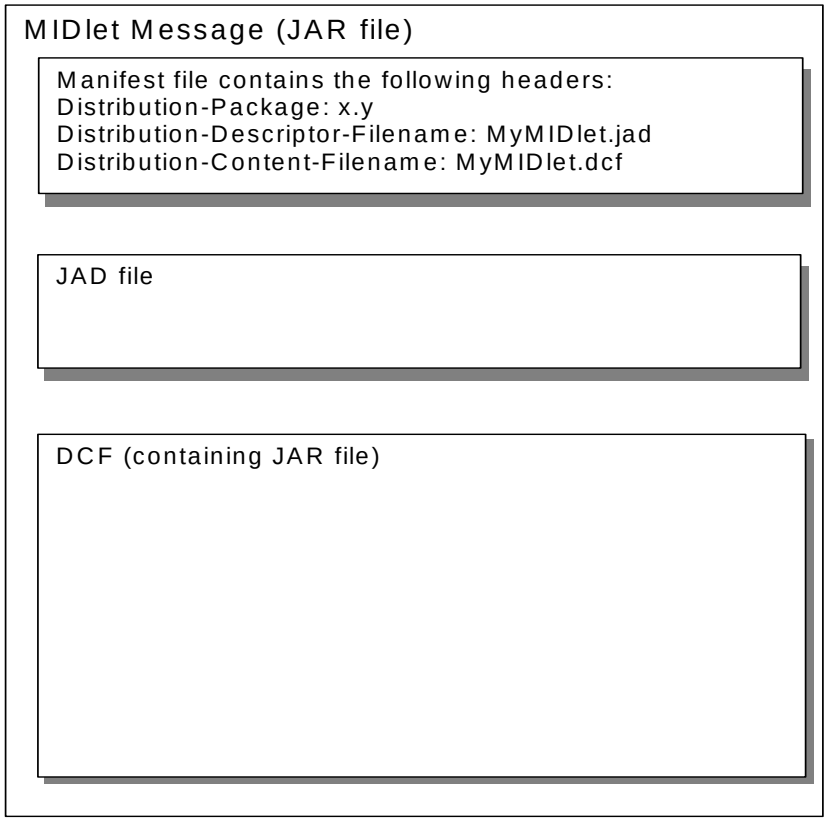

Figure 5. Overall structure of a MIDlet Message

The container JAR file is created as per the ZIP archive format specification from ANNEX A of the present document and also the JAR file specification [JAR]. Compression is not used for container JAR files.

[The container JAR file always contains exactly one JAD file and exactly one JAR file in the DCF file]5.3.r1.

[The MIME type of the container JAR file is application/java-archive]5.3.r2.

[The manifest of the JAR file (named “manifest.mf”) is placed to the META-INF directory and contains the following attributes]5.3.r3:

Attribute Name	**Attribute Description**
[Distribution-Package]5.3.r4	The version of the MIDlet Message (“1.0” for this version of the best practices document)
[Distribution-Descriptor-Filename]5.3.r5	The name of the JAD file with the full path inside the container JAR file. E.g.: “MyMIDlet.jad”
[Distribution-Content-Filename]5.3.r6	The name of the JAR file in the DCF. E.g.: “MyMIDlet.dcf”

[When creating MIDlet Messages, JAD and DCF files are placed to the root of the container JAR file, not in any folders. Names of JAD and DCF files inside the MIDlet Message can be obtained from Distribution-Descriptor-Filename and Distribution-Content-Filename attributes of the container JAR manifest]5.3.r7.

[The software that deals with unpacking of MIDlet Messages should support any order of files (JAD, DCF and manifest) inside the container JAR file, not only the order depicted in Figure 5. E.g., the manifest file can be stored the last in the container JAR file]5.3.r8.

Note, that the MIDlet Message format can also be used for forwarding of MIDlet suites that are not OMA DRM protected (if the device allows so).

Note also, the MIDlet Message format is not intended for delivery of MIDlet suites OTA.

5.4 Sending and receiving of MIDlet suites via OBEX

This section describes how OBEX protocol is used for local superdistribution of MIDlets suites.

Some OBEX headers are listed in two sections below. This does not mean that the set of headers for a particular operation is limited to the mentioned ones. Other headers can also be present when sending or receiving MIDlet suites.

5.4.1 Sending to the device

When JAR (MIDlet Message) or DCF file is sent to the device via OBEX the following rules apply:

- [OBEX Name header is used and contains the name of the file being sent
- OBEX Length header is used and contains the length of the file being sent
- OBEX Type header is used and contains the MIME type of the file being sent, i.e. *application/java-archive or application/vnd.oma.drm.content*]5.4.1.r1.

Example (example is given without OBEX encoding for clarity):

Operation	Header	Content
PUT	Name	`MyMIDlet.dcf`
	Type	`application/vnd.oma.drm.content`
	Length	`34243`
	Body	`<Body of the DCF file>`

5.4.2 Receiving by the device

The device uses OBEX protocol to receive JAR (MIDlet Message) or DCF file in the following manner:

- [Content of the OBEX Type header (if present) is used to determine the MIME type of the file being received
- Content of the OBEX Name header is used when saving the received file in the local file system]5.4.2.r1.

5.5 Superdistribution of MIDlet suites via removable mass storage devices

Some devices may support superdistribution of OMA DRM protected MIDlet suites via removable mass storage devices (RMSD). This chapter describes this type of OMA superdistribution.

[Sending and receiving of MIDlet suites via RMSD happens essentially in the same manner as when local connectivity technologies (Bluetooth, Infrared, etc) are used. The only difference is that instead of being sent between two devices a MIDlet suite (as a MIDlet message or a DCF file) is saved to and then read from the RMSD]5.5.r1

The scenario described below is possible only if both the sending device and the receiving device support the same removable storage technology.

Superdistribution happens as follows:

1. The user of the sending device selects the suite to be placed on the RMSD.
2. The device makes a decision on the format to be used for superdistribution (as described in Section 5.1.1 of the present document).
3. [The suite is placed to the RMSD (as a MIDlet Message or as a DCF file)]5.5.r2.
4. The user of the sending device passes the RMSD to the user of the receiving device.
5. [The user of the receiving device inserts the RMSD into the device and uses the device UI access the suite (as a MIDlet Message or as a DCF file)]5.5.r3.
6. [The suite is installed on the receiving device as described Section 5.2.4 of this document]5.5.r4.

5.6 SUPERDISTRIBUTION OF MIDLET SUITES VIA MMS

[If the MIDlet suite is superdistributed via MMS, it shall be in the form of MIDlet Message (if the JAD file is present) or in the form of the DCF file (if the JAD file is absent)]5.6.r1.
The exact format of the MMS message used for superdistribution is outside the scope of this document.

6.

DEVICE CAPABILITY CHECKING DURING REQUEST OF RIGHTS

One problem with OMA DRM superdistribution of OMA DRM protected MIDlet suites is portability of MIDlets. In other words, it can happen so that a sent MIDlet suite will not work on the new device, or will work improperly. This chapter describes how this problem can be solved using server-side capability checking.

When the user receives a MIDlet suite with the JAR file wrapped into the DCF (superdistribution), rights object needs to be requested. A URL of the rights issuing service from the DCF file (Rights-Issuer header of the DCF file [DCF]) is used to request rights [DRM]. [The device includes HTTP User-Agent header into this request, as described in Chapter 2 of the MIDP specification [MIDP20]]6.r1. E.g.:

GET <URL from the Rights-Issuer header of the DCF file>
User-Agent: PhoneModel/234 Profile/MIDP-2.0 Configuration/CLDC-1.1

[The server uses the value of this header to determine weather the MIDlet suite is suitable for the device. If the server concludes that the suite is suitable, it may proceed with sending of the rights object]6.r2. [If the suite is unsuitable, the server may, e.g., propose to download a proper version of the suite]6.r3 The user then can download the whole suite as described in Chapter 2 of this document.

Figure 6 illustrates the process.

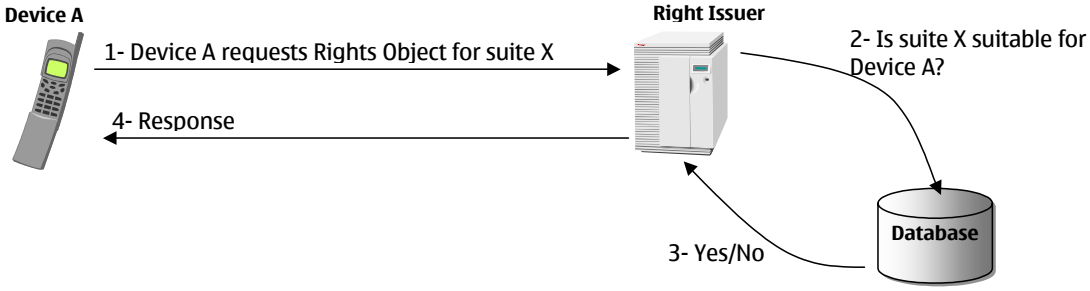

Figure 6. Server-side capability checking

Note, that the scenario shown in Figure 6 can vary due to differences in operators' infrastructure.

Note also, that [the receiving device may not contain a root certificate required to authenticate a signed superdistributed MIDlet suite. In such situations the suite installation will be interrupted with an error message on the stage where the JAD file is checked (according to the MIDP specification [MIDP20]). The rights object will never be requested] 6.r4.

ANNEX A ZIP FILE FORMAT

This chapter defines ZIP archive format that can be used to store a set of files (known as archive entries). The present ZIP format specification is based on the Info-ZIP Application Note 970311 document [ZIP] and on the ZIP Archive Resource Format document [**Error! Reference source not found.**].

Note, that all multiple-octet fields in a ZIP archive data structure use the little-endian byte order (the least byte goes first). Bits inside bytes are also in the little-endian format (i.e. the most significant bit has the highest number).

A.1 ZIP archive structure

A ZIP archive consists of the following sections (see Figure 1):

- Entry 1 … Entry N
- Directory

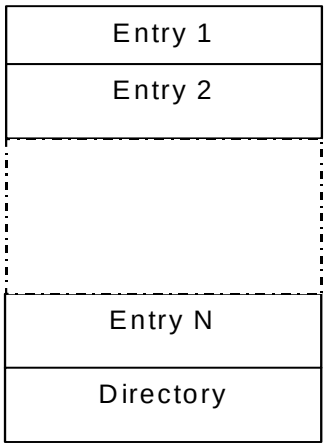

Figure 1. ZIP archive structure

A.2 Entry structure

Each ZIP archive entry is comprised of the following sections in the specified order (see Figure 2):

- Entry Header
- Entry Data

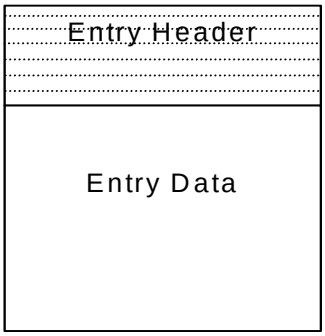

Figure 2. ZIP entry structure

A.2.1 Entry header

An entry header follows the format specified in Table 1. Each entry header is octet aligned (i.e. contains a whole number of bytes). The total size of an entry header does not exceed 65536 bytes.

Field Name	Number of bytes	Value	Comment
Marker	4	0x04034b50	A marker that indicates the start of the entry header. Constant value
VersionDecoder	2	0	A field that indicates the version of the expected decoder. Always 0
Flags	2	0	A field that is related to compression and encryption parameters. Always 0
Method	2	0	A field that specifies the compression method. Always 0
LastModTime	2	Varies*	A field that specifies the last modified time of the file placed to the entry. Bits 4-0 denote seconds divided by two (0-29), bits 10-5 denote minutes (0-59), and bits 15-11 denote hours (0-23)
LastModDate	2	Varies**	A field that specifies the last modified date of the file placed to the entry. Bits 4-0 denote day of month (1-31), bits 8-5 denote month (1-12), and bits 15-9 denote years relative to 1980
CRC32	4	Varies	A field that contains a checksum of the file stored in the entry. The checksum algorithm is as shown in *Appendix: Sample CRC Code* of RFC1952 [GZIP]
SizeCompressed	4	Varies	A field that contains the size of the file placed to the entry. The value is always

			equal to SizeUncompressed
SizeUncompressed	4	Varies	A field that contains the size of the file placed to the entry. The value is always equal to SizeCompressed
PathnameLength	2	Varies	A field that contains the length of the Pathname field in bytes
ExtensionLength	2	0	A field that contains the length of the Extension field in bytes. Extension field is always absent, therefore the value is always 0
Pathname	Varies, equal to the value of PathnameLength	Varies	The field contains the name of the file placed to the entry optionally with a relative path. The value is a non-terminated character string that conforms to the syntax described below the table

Table 1. Entry header

*. E.g. the value 0x623d represents 12:17:59:

Bytes	Least significant byte (0x3d)								Most significant byte (0x62)							
Bits meaning	Seconds (29)					Minutes (17)						Hours (12)				
Bits values	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Bits numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

**. E.g. the value 0x2f9b represents 27.12.2003:

Bytes	Least significant byte (0x9b)								Most significant byte (0x2f)							
Bits meaning	Day (27)					Month (12)				Year (23)						
Bits values	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
Bits numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

A value of the Pathname field conforms to the following syntax:

```
pathname    : [path] name
path        : pathsegment [“/” segment ]* “/”
pathsegment : anychar*
name        : anychar*
anychar     : {any non-control character except ‘/’}
```

A.2.2 Entry data

The entry data is uncompressed contents of the file placed to the entry. It is an octet-aligned sequence of bytes. The entry data starts immediately after the last byte of the entry header.

A.3 Directory structure

A ZIP directory starts immediately after the last by of the last ZIP entry and consists of the following sections in the specified order (see Figure 3):

- Directory Entry 1 … Directory Entry N
- Directory Trailer

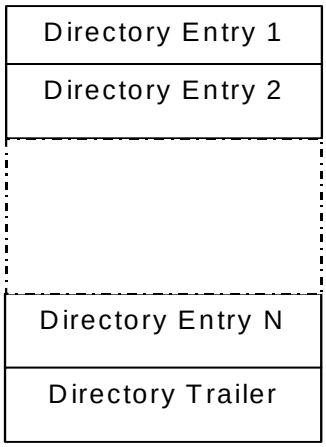

Figure 3. ZIP directory structure

A directory contains an entry for each archive entry. The order of directory entries does not necessary corresponds to the order of archive entries.

A.3.1 Directory entry

A directory entry follows the format specified in Table 2. The directory entry is octet aligned (i.e. contains a whole number of bytes).

Field Name	Number of bytes	Value	Comment
Marker	4	0x02014b50	A marker that indicates the start of the directory entry. Constant value
VersionEncoder	2	0	A field that indicates the version of the endecoder. Always 0
VersionDecoder	2	0	A field that indicates the version of the

			expected decoder. Always 0
Flags	2	0	A field, which content is related to compression and encryption parameters. Always 0
Method	2	0	A field that specifies the compression method. Always 0
LastModTime	2	Varies (see similar field in Table 1), but always the same as the corresponding field in the entry header	A field that specifies the last modified time of the file placed to the entry. Bits 4-0 denote seconds divided by two (0-29), bits 10-5 denote minutes (0-59), and bits 15-11 denote hours (0-23)
LastModDate	2	Varies (see similar field in Table 1), but always the same as the corresponding field in the entry header	A field that specifies the last modified date of the file placed to the entry. Bits 4-0 denote day of month (1-31), bits 8-5 denote month (1-12), and bits 15-9 denote years relative to 1980
CRC32	4	Varies	A field that contains a checksum of the file stored in the entry. The checksum algorithm is as shown in *Appendix: Sample CRC Code* of RFC1952 [GZIP]
SizeCompressed	4	Varies	A field that contains the size (in bytes) of the file placed to the entry. The value is always equal to SizeUncompressed
SizeUncompressed	4	Varies	A field that contains the size (in bytes) of the file placed to the entry. The value is always equal to SizeCompressed
PathnameLength	2	Varies	A field that contains the length of the Pathname field in bytes
ExtensionLength	2	0	A field that contains the length of the Extension field in bytes. Extension field is always absent, therefore the value is always 0
CommentLength	2	0	A field that contains the length of the Comment field in bytes. Extension field is always absent, therefore the value is always 0
Segment	2	0	This field contains the number of the archive segment (volume) within which

			this archive entry starts. Multiple volumes are not supported, therefore the value is always 0
AttributesInternal	2	0	The value of this field indicates the nature of the data in the entry – textual or binary. Always set to 0 (binary)
AttributesExternal	4	0	Always set to 0
Offset	4	Varies	The field contains the offset from the start of the archive to the entry's header. E.g., for the first entry of the archive the value is 0, for the second entry – the size of the first archive entry, etc.
Pathname	Varies, equal to the value of PathnameLength	Varies	The field contains the name of the file placed to the entry optionally with a relative path. The value is a non-terminated character string that conforms to the syntax described in Section A.2.1 after Table 1

Table 2. Directory entry

A.3.2 Directory trailer

A directory trailer follows the format specified in Table 3. The directory trailer is octet aligned and follows the last byte of the last directory entry.

Field Name	Number of bytes	Value	Comment
Marker	4	0x06054b50	A marker that indicates the start of the directory trailer. Constant value
SegmentTrailer	2	0	This field contains the number of the archive segment (volume) within which the directory trailer starts. Multiple volumes are not supported, therefore the value is always 0
Segment Directory	2	0	This field contains the number of the archive segment (volume) within which the archive directory starts. Multiple volumes are not supported, therefore the value is always 0
EntriesLast	2	Varies, always equal to the value of Entries	This field contains the number of directory entries in the present archive segment (volume). Multiple volumes are not supported, therefore the value is always equal to the value of the Entries field

Entries	2	Varies, always equal to the value of EntriesLast	This field contains the total number of directory entries
Size	4	Varies	A field that contains the size of the archive directory (without the directory trailer!) in bytes
Offset	4	Varies	The offset in bytes from the start of the ZIP archive to the beginning of the directory (i.e. to the first directory entry)
CommentLength	2	0	A field that contains the length of the Comment field in bytes. Comment field is always absent, therefore the value is always 0

Table 3. ZIP directory trailer

A.4 Example

This section contains an example of a ZIP archive. Three files are placed to the archive according to the present specification. File names and content are as follows:

File name (with relative path)	**File content**
META-INF/manifest.mf	Contents of manifest.mf file
midlet.jad	Contents of midlet.jad file
midlet.jar	Contents of midlet.jar file

Resulting ZIP archive looks as follows:

ZIP archive contents, from left to right, from top to bottom, hexadecimal notation	**Comment**
50 4b 03 04	Marker of manifest.mf file archive entry
00 00 00 00 00 00	VersionDecoder, Flags, Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
f0 e7 5e 44	CRC32 of manifest.mf

1c 00 00 00 1c 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value
14 00	PathnameLength field. Contains the length (20) of the following string “META-INF/manifest.mf”
00 00	ExtensionLength field. Always 0
4d 45 54 41 2d 49 4e 46 2f 6d 61 6e 69 66 65 73 74 2e 6d 66	Pathname field. “META-INF/manifest.mf” string in UTF-8 encoding
43 6f 6e 74 65 6e 74 73 20 6f 66 20 6d 61 6e 69 66 65 73 74 2e 6d 66 20 66 69 6c 65	Contents of manifest.mf file (“Contents of manifest.mf file”).
50 4b 03 04	Marker of midlet.jad file archive entry
00 00 00 00 00 00	VersionDecoder, Flags, Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
4c 00 d0 8c	CRC32 of midlet.jad
1b 00 00 00 1b 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value
0A 00	PathnameLength field. Contains the length (10) of the following string “midlet.jad”
00 00	ExtensionLength field. Always 0
6d 69 64 6c 65 74 2e 6a 61 64	Pathname field. “midlet.jad” string in UTF-8 encoding
43 6f 6e 74 65 6e 74 73 20 6f 66 20 6d 69 64 6c 65 74 2e 6a 61 64 20 66 69 6c 65	Contents of midlet.jad file (“Contents of midlet.jad file”).
50 4b 03 04	Marker of midlet.jar file archive entry
00 00 00 00 00 00	VersionDecoder, Flags, Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
ca e1 5f 59	CRC32 of midlet.jar
1b 00 00 00 1b 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value
0A 00	PathnameLength field. Contains the length (10) of the following string “midlet.jar”
00 00	ExtensionLength field. Always 0
6d 69 64 6c 65 74 2e 6a 61 72	Pathname field. “midlet.jar” string in UTF-8 encoding
43 6f 6e 74 65 6e 74 73 20 6f 66 20 6d 69 64 6c 65 74 2e 6a 61 72 20 66 69 6c 65	Contents of midlet.jar file (“Contents of midlet.jar file”).
50 4b 01 02	Marker of manifest.mf file directory entry

00 00 00 00 00 00 00 00	VersionEncoder, VersionDecoder, Flags and Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
f0 e7 5e 44	CRC32 of manifest.mf
1c 00 00 00 1c 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value
14 00	PathnameLength field. Contains the length (20) of the following string “META-INF/manifest.mf”
00 00 00 00 00 00 00 00 00 00 00 00	ExtensionLength, CommentLength, Segment, AttributesInternal and AttributesExternal fields. Always 0
00 00 00 00	Offset field. Has value 0, as the archive entry that contains manifest.mf file is first in the archive
4d 45 54 41 2d 49 4e 46 2f 6d 61 6e 69 66 65 73 74 2e 6d 66	Pathname field. “META-INF/manifest.mf” string in UTF-8 encoding
50 4b 01 02	Marker of midlet.jad file directory entry
00 00 00 00 00 00 00 00	VersionEncoder, VersionDecoder, Flags and Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
4c 00 d0 8c	CRC32 of midlet.jad
1b 00 00 00 1b 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value
0A 00	PathnameLength field. Contains the length (10) of the following string “midlet.jad”
00 00 00 00 00 00 00 00 00 00 00 00	ExtensionLength, CommentLength, Segment, AttributesInternal and AttributesExternal fields. Always 0
4e 00 00 00	Offset from the start of the archive to the archive entry that contains midlet.jad file
6d 69 64 6c 65 74 2e 6a 61 64	Pathname field. “midlet.jad” string in UTF-8 encoding
50 4b 01 02	Marker of midlet.jar file directory entry
00 00 00 00 00 00 00 00	VersionEncoder, VersionDecoder, Flags and Method fields. Always 0
26 63	LastModTime field (12:25:12)
95 2f	LastModDate field (21/12/2003)
ca e1 5f 59	CRC32 of midlet.jar
1b 00 00 00 1b 00 00 00	SizeCompressed and SizeUncompressed fields. They always have the same value

0A 00	PathnameLength field. Contains the length (10) of the following string “midlet.jar”
00 00 00 00 00 00 00 00 00 00 00 00	ExtensionLength, CommentLength, Segment, AttributesInternal and AttributesExternal fields. Always 0
91 00 00 00	Offset from the start of the archive to the archive entry that contains midlet.jar file
6d 69 64 6c 65 74 2e 6a 61 72	Pathname field. “midlet.jad” string in UTF-8 encoding
50 4b 05 06	Marker of directory trailer
00 00 00 00	SegmentTrailer and SegmentDirectory fileds. Always 0
03 00 03 00	EntriesLast and Entries fields. Always have the same value equal to the total number of entries in the archive. In this case 3
b2 00 00 00	The size of the archive directory (without the directory trailer!) in bytes. In this case 178
D4 00 00 00	Offset in bytes from the beginning of the archive to the directory. In this case 212
00 00	CommentLength field. Always 0