Over The Air User Initiated Provisioning Recommended Practice

for the Mobile Information Device Profile

Version 1.0

May 7, 2001

Please send comments to *midp-feedback@risc.sps.mot.com*

Preface

This document, *Over The Air User Initiated Provisioning Recommended Practice,* is for the *Mobile Information Device Profile* (MIDP) specification version 1.0 which can be found at http://jcp.org/jsr/detail/37.jsp.

The terminology used herein is defined by that specification except where noted.

Revision History

TABLE P-1 Revision History

Date	Version	Description
May 7, 2001	1.0	Incorporated editorial changes based on industry feedback

Who Should Use This Specification

The audience for this document is the Java Community Process (JCP) expert group that defined the MIDP, implementors of the MIDP, application developers using the MIDP, service providers deploying MIDP applications, and wireless operators deploying the infrastructure to support MIDP devices.

How This Specification Is Organized

The topics in this specification are organized according to the following chapters:

"Questions and Answers," presents a summary of the issues and feedback received in the initial public review of this document.

Appendix A, "Over The Air User Initiated Provisioning," defines how MIDP applications should be distributed to wireless devices.

Appendix B, "MIDP Provisioning and Networking in the WAP June2000 Environment," describes the specific requirements for deploying MIDP applications via a proxied WAP Gateway.

References

1. Connected, Limited Device Configuration (CLDC)
   http://jcp.org/jsr/detail/30.jsp
2. Mobile Information Device Profile (MIDP)
   http://jcp.org/jsr/detail/37.jsp
3. HTTP 1.1 Specification
   http://www.ietf.org/rfc/rfc2616.txt
4. HTTP Authentication: Basic and Digest Access Authentication
   http://www.ietf.org/rfc/rfc2617.txt
5. HTTP State Management Mechanism (Cookies)
   http://www.ietf.org/rfc/rfc2109.txt

6. Java(tm) Servlet 2.3 Specification
   http://jcp.org/jsr/detail/53.jsp

7. The following words, as defined in http://www.ietf.org/rfc/rfc2119.txt, indicate requirement levels: “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL”.

Questions and Answers

Note – Throughout this document, the term *MIDlet suite* is used to identify the unit of installation. This term is used to be consistent with the MIDP 1.0 specification’s language. In addition, the CLDC and MIDP specifications require support of compressed JAR files as an installation file format, but do not prohibit an implementation from having additional installation capabilities.Since these additional capabilities are not part of the CLDC and MIDP specifications, this document addresses only the case in which a MIDlet suite is totally contained in a compressed JAR file, and that JAR file is what is being transported over-the-air to the device.

These are some of the questions and issues, along with the answers, that were raised during the review period of this document.

1. What is the scope and lifetime of the user name and password acquired when processing HTTP response codes of 401 and 407? (*i.e.*, as needed for Basic Authentication)

   It is up to the device to specify and implement mechanisms that prompt the user for user names and passwords and to also define how long such information is retained.

2. Should a new MIME type be introduced for the JAR file containing a MIDlet suite?

   The IANA registers MIME types for specific *file formats*. Since the JAR file containing a MIDlet suite conforms to the existing specification for a JAR file, there is no additional requirement for creating a new MIME type. See RFC2048 for the requirements and procedures for creating new MIME types.

3. How should session information be included in requests for a MIDlet suite download?

   The current cookie mechanism defined in RFC2109 SHOULD be used as the session key for the MIDlet suite installation. The implementation limits defined in RFC2109 suggest a length of 4096 bytes for cookies. For installation of MIDlet

suites, the `Set-Cookie` headers should contain no more than 256 bytes. The cookie only needs to be kept for the duration of the installation process and MUST be discarded when installation is complete.

4. Should the `MIDlet-Install-Notify` URL be moved to the manifest within the JAR?

   The `MIDlet-Install-Notify` attribute needs to be in the application descriptor so that installation status reporting can occur even in the case where the MIDlet suite cannot be retrieved (*e.g,* if the JAR-file URL is incorrect). The `MIDlet-Install-Notify` attribute may also appear in the manifest using the same pattern used for other MIDlet attributes.

5. An installation status code of 903 (Loss of Service) was defined but never used.

   Loss of service should be returned if the installation failed due to a service interruption. Reporting this status is optional, because the network is not likely to be available to report the status.

6. Should the `If-Modified-Since` header be included in requests for the MIDlet suite?

   No. The use of the `If-Modified-Since` header should not be used for version management since the MIDP 1.0 specification already defines MIDlet suite version management.

7. Should the `Accept` header include the type */* when requesting the JAR?

   The `Accept` header should only be used when specific types are being requested. Overuse of “Accept: */*” can lead to the device receiving data that it can not process. In the case of JAR files, the IANA approved MIME type is “application/java-archive”. The `Accept` header MUST be included for implementations proxied via a WAP gateway.

8. Should the OTA Provisioning specification include unambiguous identification of the device to the network during provisioning?

   There is considerable sensitivity from the privacy, legal, and social aspects to providing information that identifies the customer without their explicit consent to any application that is installed on a device. The provision of the basic authentication scheme and session cookie incorporated into this document are sufficient to identify the device and user making an installation request. This document cannot require information to be revealed that would violate these constraints.

9. When downloading a MIDlet suite, should the headers in the “Device Identification and Request Headers” on page 17” be used?

The application descriptor MUST uniquely identify the specific MIDlet suite to be downloaded so that the device can determine whether it should perform an upgrade or a new installation.

10. When the application descriptor is downloaded, what character set is used to convert it to Unicode for processing?

    The default character set specified for the MIME type "`text/vnd.sun.j2me.app-descriptor`" is `UTF-8`. If the device supports other character sets, the appropriate `Accept-Charset` header should be included in the request, and the content should be decoded based on the `charset` parameter returned on the `Content-Type` header (*e.g.,* `Content-Type: text/vnd.sun.j2me.app-descriptor; charset=UTF-8)`.

11. What is the requirement for checking the size of the JAR file listed in the descriptor with actual MIDlet suite size?

    It is becoming increasing clear from comments in the feedback and experience with implementations that the size indicated in the application descriptor attribute `MIDlet-Jar-Size` can not be used as the memory requirement on the device. The actual memory required on the device may be larger or smaller depending on the device. This value can only be used as a hint to the device and the user. The device should provide the user with practical information that will allow them to determine how much space is needed to install and run the application.

12. What size of MIDlet suite can be guaranteed to work when requested by a MIDP client device?

    There are no prescribed limits, either minimum or maximum, related to the sizes of application descriptors or MIDlet suites.

13. Are there special requirements that must be satisfied when OTA provisioning is deployed using WAP devices and infrastructure?

    Appendix B, "MIDP Provisioning and Networking in the WAP June2000 Environment" describes the goals and specifications for performing OTA provisioning in the WAP network environment specifically to promote interoperability between devices and WAP June2000 Environments.

14. When an installation status report cannot be posted to the server is the application installation considered successful?

    Yes, since wireless devices may lose network connectivity due to roaming, radio interference, *etc*.

15. When requesting a MIDlet suite, how would the device request a compressed JAR?

    The CLDC specification states that all CLDC-compliant devices must accept a

compressed JAR file; therefore, it is assumed that all JAR files deployed will be compressed. Note that this does not preclude a device from supporting other installation file formats; however, the description of such alternate formats is beyond the scope of this document.

16. Should different errors codes be returned in the installation status report to indicate more precisely the type of failure (*e.g.*, to distinguish between insufficient memory conditions for cookie storage as opposed to MIDlet suite installation storage)?

    No. There are many possible reasons for failure for OTA provisioning. Rather than try to enumerate them all at present, this document recommends reporting the general condition. More specific error codes may be added in the future.

APPENDIX **A**

Over The Air User Initiated Provisioning

A.1 Overview and Goals

The purpose of this recommended practice is to describe how MIDlet suites can be deployed Over-The-Air (OTA), and the functions the device should provide to support these deployments. Following these recommendations will help ensure interoperability between devices from all manufacturers and provide guidance to cellular operators deploying MIDP 1.0 devices. The aim of the document is to provide a de facto standard that will support full, interoperable use for MIDP 1.0 devices and be a foundation for subsequent MIDP specifications.

Devices are expected to provide mechanisms that allow users to discover MIDlet suites that can be loaded into the device. In some cases, discovery will be via the device's resident browser (*e.g.*, WAP™). In other cases, it may be a resident application written specifically to identify MIDlet suites for the user to download. Throughout this document, this functionality is referred to as the *discovery application*, or DA.

During the download and installation process, the user MUST be given opportunities to control the download, determine which versions of software are installed, remove MIDlet suites once downloaded, and obtain information about the MIDlet suite(s) on the device. For any operation, the user SHOULD be informed of progress and given an opportunity to cancel the activity. The user retains control of the resources used by MIDlet suites on the device and MAY delete or install MIDlet suites.

The device MUST make the MIDlet(s) in the MIDlet suite available to be run by the user. When multiple MIDlets are contained in a MIDlet suite, the user MAY need to be aware that there is more than one. When a MIDlet suite is installed, a notice SHOULD be sent to the server to indicate installation has completed. The server MAY use this information for accounting or for other customer service needs. The device MAY run the MIDlet immediately at the user's option and SHOULD warn the user during removal of any special circumstances regarding the deletion of the MIDlet suite. For example, if a MIDlet suite contains multiple MIDlets, it SHOULD be clear that all of the MIDlets are being removed.

The term Application Management Software (AMS) is a generic term used to describe the software on the device that manages the downloading and lifecycle of MIDlets. This term does not imply any specific implementation and is used for convenience only. In some implementations, the term Java Application Manager (JAM) is used interchangeably.

This document describes the general functional requirements on the device and the functions supporting the MIDlet suite lifecycle. The lifecycle of a MIDlet suite consists of discovery, installation, update, invocation and removal. Descriptions are included for additional application descriptor attributes and mechanisms that identify the device type and characteristics to the server and services providing MIDlet suites.

A.2 Functional Requirements

A MIDP-compliant device MUST be capable of:

- Browsing, or otherwise locating MIDlet suite application descriptors in the network.
- Transferring a MIDlet suite and its associated application descriptor to the device from a server using HTTP 1.1 or a session protocol that implements the HTTP 1.1 functionality (including the header and entity fields) as required in this document.
- Responding to a server 401 (unauthorized) or 407 (proxy authentication required) response to a HTTP request by asking the user for a user name and password and re-sending the HTTP request with the credentials supplied. The device MUST be able to support at least the RFC2617 Basic Authentication Scheme.
- Installing the MIDlet suite on the device
- Invoking MIDlets
- Allowing the user to delete MIDlet suites stored on the device Single MIDlets cannot be deleted since the MIDlet suite is the unit of transfer and installation.

A.3 Application Discovery

Application discovery is the process by which a user uses the device to locate a MIDlet suite. User-initiated discovery and installation of MIDlet suites MUST be supported in the following high-level manner:

- While using the DA, the user is presented with a link to an application descriptor.
- The user selects the link to begin the installation process
- The descriptor is transferred to the device.
- The descriptor is used by the device’s AMS to start installation

Using the DA, the user SHOULD be able to access a network location and see a description of the MIDlet suite along with a link that when selected, initiates the installation of the MIDlet suite. This link MUST be a reference to an application descriptor as described in the MIDP 1.0 specification. Once the link has been selected, the server MUST indicate that the

data being transferred (*i.e.*, the application descriptor) has a MIME type of "`text/vnd.sun.j2me.app-descriptor`", and the extension of the transferred file MUST be "`jad`".

After completing this transfer, the application descriptor, is passed to the AMS on the device to start the installation process. The application descriptor is used by the AMS to determine if the associated MIDlet suite can be successfully installed and executed on the device. If not, the user MUST be notified of the conditions which prevent its installation. The user SHOULD be informed of unusual conditions as early as possible to minimize wasted time and network bandwidth. The request-header attributes described in "Device Identification and Request Headers" on page 17 SHOULD be used when retrieving the application descriptor.

To aid the user in selecting MIDlet suites appropriate for the device, the DA SHOULD be able to inform the network server about the device's capabilities. In many cases, a DA will already have identified the device type to the server by means consistent with its network connection and markup language.

A.4 Application Installation

Application installation is the process by which a MIDlet suite is downloaded onto the device and made available to the user. Application installation MUST be supported. The network supporting the devices MUST be able to support this requirement as well including any proxies and origin servers that are used during provisioning. To install a MIDlet suite, the AMS performs the following series of steps and checks and provides the user with feedback about the progress. The user MUST be given a way to cancel between any interruptible steps. Interrupting installation MUST leave the device and MIDlet suites in working order in the same state as before the installation began:

1. The application descriptor MUST be converted from its transport format to the Unicode-encoding that is specified by the MIDP specification before it can be used. The default character set specified for the MIME type "`text/vnd.sun.j2me.app-descriptor`" is "`UTF-8`." If the device supports other character sets, the appropriate `Accept-Charset` header SHOULD be included in the request, and the content SHOULD be decoded based on the `charset` attribute returned on the `Content-Type` header. If no `charset` parameter is defined, the encoding defaults to "`UTF-8`," and it SHOULD be decoded accordingly. The attributes in the descriptor MUST be formatted according to the syntax in the MIDP 1.0 specification. If not, then an error code of 906 MUST be returned in the status report.

2. Using the information in the application descriptor including the vendor, name, version, and size attributes, the user SHOULD be given a chance to confirm that they want to install the MIDlet suite. Situations such as trying to install an older version, or installing the same version, SHOULD be brought to the user's attention. Conditions that can prevent the successful installation and execution of the MIDlet suite SHOULD be identified, and the user notified. For example, if it is

known that insufficient space is available, the software SHOULD aid the user in reviewing memory usage and freeing sufficient memory for installation of the new MIDlet suite.

3. If the MIDlet suite is already installed on the device, it SHOULD be treated as a potential upgrade. As part of the updating process, the persistent data of the MIDlet suite SHOULD be preserved for use by the updated MIDlet suite. See “Device Identification and Request Headers” on page 17 for the attributes that apply to upgrades. If the current version is already installed, no further installation work is required, and the MIDlet suite is already available for execution by the user.

4. Using the URL from the application descriptor, the device initiates the download of the MIDlet suite via HTTP. The request for the MIDlet suite MUST be for exactly the URL specified in the descriptor, and additional headers are unnecessary. If a `cookie` was returned with the application descriptor, it MUST be included in the subsequent request for the MIDlet suite.

5. If the server or proxy responds to the request for the MIDlet suite with a 401 (`Unauthorized`) or 407 (`Proxy Authentication Required`), the device SHOULD re-send the request with the user-supplied credentials in an `Authorization` or `Proxy-Authorization` 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 device MUST be able to support at least the *Basic Authentication Scheme* as described in RFC2617.

6. If the response contains a “`Set-Cookie`” header as defined in RFC2109, it SHOULD be treated as a session token, and the `Cookie` attribute names and values remembered so that the `cookie` can be used in the `Installation Status Report,` or in a re-send of the request with authentication credentials. The implementation limits defined in RFC2109 require only temporary storage of 4096 bytes. For MIDP devices, `Set-Cookie` headers MUST contain no more than 256 bytes. The device MUST be able to store a single `Cookie` for the duration of the installation process. If multiple `Set-Cookie` headers are received, only the first SHOULD be used. If insufficient memory is available, the entire `Cookie` MUST be discarded and ignored.

7. The MIDlet suite and the headers that are received MUST be checked to verify that the retrieved MIDlet suite is valid and can be installed on the device. The user MUST be alerted to at least the following specific conditions that prevent installation:

   a. If there is insufficient memory to store the MIDlet suite on the device, the device MUST return `Error Code 901` in the Status Report.

   b. If the received JAR file size does not match the size specified in the application descriptor, the device MUST return `Error Code 904` in the Status Report.

c. If the mandatory attributes in the descriptor "`MIDlet-name`", "`MIDlet-Version`", and "`MIDlet-vendor`" do not match those in the manifest, the device MUST return `Error Code 905` in the Status Report.

d. IIf the network service is lost during installation, an `Error Code 903` SHOULD be used in a Status Report if possible (it may be impossible to deliver the status report due to the network-service outage).

8. The MIDlets contained in the MIDlet suite MUST be installed and be made available to be run by the user via the device's MIDlet selection mechanism.

9. Installation is complete when the MIDlet suite has been made available on the device, or an unrecoverable failure has occurred. In either case, the status MUST be reported as described in "Installation Status Reports" on page 15.

A.5 Installation Status Reports

The success or failure of the installation or upgrade of a MIDlet suite is of interest to the service providing the MIDlet suite. The service MAY specify in the application descriptor a URL which is to be used to report install status. If the device can not establish a connection to the URL, the MIDlet suite MUST still be enabled.

The installation status is reported by means of an HTTP POST request to URL that is the value of the `MIDlet-Install-Notify` attribute. The only protocol that MUST be supported is "http://". Any other protocol MAY be ignored by the device.

If a `cookie` was returned when the application descriptor or MIDlet suite was retrieved, and the domain and path in the `cookie` match the URL in the `MIDlet-Install-Notify` attribute, then the `cookie` MUST be included in the request headers so that the server can identify the installation which has been completed.

The content of the body of the POST request MUST include on the first line a status code and status message. See "Install Status Codes and Message" for list of valid status message codes and messages.

In response to an installation status report, the server need only reply with a "`200 OK`" response. No content SHOULD be returned to the device and, if any is sent, it MUST be ignored. If a request brings no response, the request may be retried, but it SHOULD NOT be retried if any response is received. The server SHOULD include a `Set-Cookie` header that

is exactly the same as the one used for installation session identification, but with the attribute with `Max-Age=0` to confirm that the `cookie` MUST be discarded. As an example, please see “Example: Install Status via HTTP Post Request” on page 19.

TABLE 1-1 Install Status Codes and Message

Status Code	Status Message
900	Success
901	Insufficient Memory
902	User Cancelled
903	Loss of Service
904	JAR size mismatch
905	Attribute Mismatch
906	Invalid Descriptor

A.6 Application Update

Update of MIDlet suites in devices MUST be supported. The scenarios below describe only two of the possible situations in which the update may occur.

In the first scenario, a user attempts to install a MIDlet suite found using the DA, and the MIDlet suite is already installed on the device. In this case, the device SHOULD notify the user if it is an older or newer version and confirm before proceeding. If the descriptor was not loaded from the same network domain as the currently installed MIDlet suite, there is the risk that it may not actually be an upgrade to the MIDlet suite. If it is not actually an upgrade, there is the possibility of exposing the persistent storage of the original MIDlet suite to the new MIDlet suite. This represents a security risk and the device SHOULD warn the user of this situation and MAY require the original MIDlet suite to be removed before installing the new one.

In the second scenario, the device’s AMS MAY provide a mechanism to upgrade a MIDlet suite that is already installed. To upgrade, the application descriptor SHOULD be fetched from the network and used to as input to the installation process already described. The normal installation process already checks to see if the descriptor is for newer version and upgrade the MIDlet suite accordingly.

The persistent data of the MIDlet suite SHOULD be preserved for use by the updated MIDlet suite. The format, contents and versioning of the RecordStores is the responsibility of the individual MIDlet suite.

A.7 Application Execution

When the user selects a MIDlet to be run, the device SHOULD invoke the MIDlet with the CLDC and MIDP classes as required by the MIDP specification. If multiple MIDlets are present the MIDlet suite the user interface MUST allow the user to select which one to run.

A.8 Application Removal

When a MIDlet suite is to be removed from the device, the user SHOULD be prompted to confirm that the MIDlet suite should be removed. If the application descriptor includes the attribute MIDlet-Delete-Confirm, its value SHOULD be included in the prompt. This will allow the MIDlet suite provider to highlight any specific conditions that might arise if the MIDlet suite were to be removed.

The MIDlet suite MAY contain multiple MIDlets, and the user SHOULD be made aware that all of the MIDlets and associated RecordStores are being removed.

A.9 Additional Descriptor Attributes

The following additional attributes are defined in the application descriptor. Each may appear only once the descriptor.

TABLE 1-1 MIDlet Attributes

Attribute Name	Attribute Description
MIDlet-Delete-Confirm	A text message to be provided to the user when prompted to confirm deletion of this MIDlet suite
MIDlet-Install-Notify	The URL to which a POST request is used to confirm successful installation of this MIDlet suite

A.10 Device Identification and Request Headers

The discovery of a MIDlet suite via the DA can be customized or augmented by the device sending information about itself to the server.

During the download of a MIDlet suite, a device SHOULD identify its characteristics and the type of the content being requested as completely as possible to the server. The HTTP request-headers used to fetch the content SHOULD include, as appropriate, `User-Agent`, `Accept-Language`, and `Accept`. Servers SHOULD use this additional information to select the appropriate application descriptor for the device.

User-Agent Product Tokens

The MIDP 1.0 specification identifies HTTP `User-Agent` request headers to identify the client to the server. RFC2616 specifies a format for product tokens such as:

```
"User-Agent" ":" 1*(product "/" product-version)
```

The product tokens used to identify the device as supporting CLDC and MIDP are specified in Table 6.1 of the MIDP specification.

In addition, the device SHOULD further identify itself by adding a device-specific product token to the `User-Agent` header as defined by RFC2616. The device-identifying token SHOULD be the first token. The product-token and product-version values are specific to each device and are outside of the scope of this specification.

Accept-Language Header

The device MAY supply the `Accept-Language` request-header as specified in RFC2616 to indicate the language that is in use on the device.

Accept Header

The `Accept` HTTP header is used to indicate the type of content being requested. If supplied when requesting MIDlet suites, this header SHOULD include `application/java` and `application/java-archive`. For retrieving application descriptors, this header should include `text/vnd.sun.j2me.app-descriptor.`

Example: HTTP Request for Application Descriptor

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

```
GET http://host.foo.bar/app-dir/game.jad HTTP/1.1
Host: host.foo.bar
Accept: text/vnd.sun.j2me.app-descriptor
User-Agent: CoolPhone/1.4 Profile/MIDP-1.0 Configuration/CLDC-1.0
Accept-Language: en-US, fi, fr
Accept-Charset: utf-8
```

The response headers from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Set-Cookie: Name=”abc”; Domain=”.foo.bar”; Path=”/app-dir”; \
        JSESSIONID=”123”; VERSION=”1”
Content-Length: 2345
Content-Type: text/vnd.sun.j2me.app-descriptor; charset=utf-8
```

Example: HTTP Request to Install/Update a MIDlet Suite

When requesting the download of a MIDlet suite file, the request headers might look as follows:

```
GET http://host.foo.bar/app-dir/game.jar HTTP/1.1
Host: host.foo.bar
Cookie: Name=”abc”; Domain=”.foo.bar”; Path=”/app-dir”; \
        JSESSIONID=”123”; VERSION=”1”
Accept: application/java, application/java-archive
```

The response headers from server might look as follows:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Content-Length: 25432
Content-Type: application/java-archive
Cookie: Name=”abc”; Domain=”.foo.bar”; Path=”/app-dir”; \
        JSESSIONID=”123”; VERSION=”1”
```

Example: Install Status via HTTP Post Request

For example, installing a MIDlet suite with an application descriptor given below:

```
...
MIDlet-Install-Notify: http://foo.bar.com/status
...
```

After a successful install of the MIDlet suite, the following would be posted:

```
POST http://foo.bar.com/status HTTP/1.1
Host: foo.bar.com
Cookie: Name=”abc”; Domain=”.foo.bar”; Path=”/app-dir”; \
        JSESSIONID=”123”; VERSION=”1”
Content-Length: 13

900 Success
```

The response from the server might be:

```
HTTP/1.1 200 OK
Server: CoolServer/1.3.12
Cookie: Name=”abc”; Domain=”.foo.bar”; Path=”/app-dir”; \
         JSESSIONID=”123”; VERSION=”1”; Max-Age=”0”
```

APPENDIX **B**

MIDP Provisioning and Networking in the WAP June2000 Environment

B.1 Purpose of This Document

The purpose of this section is to complement [*OTA*] and [*MIDP*] by providing requirements and recommendations specific to MIDP Over The Air Provisioning and MIDlet networking in the WAP June2000 environment. Future WAP developments will be addressed in future versions of the MIDP. Following these recommendations will help ensure interoperability between different WAP elements from all manufacturers. It will also provide guidance to cellular operators in deploying MIDP 1.0 services when provisioning is performed via a browser using the WAP protocol stack, as well as to MIDlet developers in creating MIDlets that function optimally when the transport is via WSP. The aim of the section is to provide a de-facto standard to WAP environments that will support full interoperable use for MIDP 1.0 devices and be a foundation for subsequent MIDP specifications.

B.2 Overview

MIDlet suites are downloaded using HTTP from a provisioning server (possibly via a gateway in between). Also, the MIDP library MUST support network access in the form of the HTTP/1.1 protocol. More information can be found in MIDP specification [*MIDP*].

Depending on the end-user device and the wireless network, the communication between the end-user device and provisioning server MAY happen with the HTTP protocol end-to-end, or the end-user device MAY use another protocol and a gateway that converts this protocol to HTTP. The provisioning server needs to only support HTTP in any case (unless there are other reasons for the same service provider to operate the protocol gateway as well). In WAP June2000 environments, there is always a WAP gateway between the terminal and the provisioning server to translate between the WSP protocol used to communicate with the device and TCP/IP used to communicate with the server.

There are essentially two basic interfaces that need to be considered:

- the interface from the end-user device to the network

- the interface from the provisioning server to the network

The latter of these interfaces will always be HTTP carried as usual over TCP/IP.

For the former interface, this document describes one of the two basic cases:

- the end-user device uses a browser using the WAP protocol stack and
- the WSP protocol is used for communication between the terminal and the WAP gateway.

It is not within the scope of this appendix to describe the case where the end-user device uses directly HTTP over TCP/IP for either application discovery or MIDlet network support.

When the end-user device uses a browser using the WAP protocol stack and has the WAP transport protocols, WSP MAY be used instead of HTTP in the end-user device. Only connection-oriented WSP and only the following WAP protocol stack configurations and bearers are supported:

- WAP/UDP/IPv4/PPP/CSD
- WAP/UDP/IPv4/GPRS

Where WAP can be either:

- WSP/WTP/WTLS, or
- WSP/WTP

(The other bearers in [*WAP_WDPS*], such as SMS- or USSD -based bearers are not supported). These restrictions are made in order to achieve maximal interoperability in MIDP provisioning in WAP environments.

Depending on the wireless network and the capabilities of the end-user device, different mechanisms for obtaining the IP connection are used. These mechanisms and their required configurations are outside the scope of this appendix.

B.3 Terminal Requirements and Recommendations

This section lists the requirements and recommendations related to the WAP terminals. In the case of common requirements to both terminals and gateways, the requirements and recommendations are listed in both terminal and gateway sections.

WAP terminals used MUST be WAP June2000 conformant [*WAP_JUNE2000*].

Specifically, the following issues are critical:

- JAD and JAR MIME-types [*MIDP*] MUST be supported.
- HTTP authentication (server responses 401 and 407) MUST be supported.
- POST-messages from the terminal to provisioning server MUST be supported.

Requirements and recommendations in addition to those in the WAP

Specifications:

In the case where the HTTP connections are implemented over WSP, the system implementation of HTTP MUST add the request header "`Accept: */*`" to GET and POST requests when a MIDlet creates an HTTP-request, but does not include a non-empty accept-header in the request. This ensures that the WAP Gateway will always have an explicit set of types and will pass the requested data. This is conceptually the same as leaving accept-header out from an HTTP-request on other transports. If the MIDlet sets a non-empty accept-header for its HTTP-request, no change is made (MIDlets own accept-field is the only one sent).

B.4 Gateway Requirements and Recommendations

This section lists the requirements and recommendations related to the WAP gateways. The purpose of presenting these issues here is to make sure that they are taken into consideration when WAP-based MIDlet provisioning is considered.

WAP gateways used MUST be WAP June2000 conformant [*WAP_JUNE2000*].

Specifically, the following issues are critical:

- JAD and JAR MIME-types [*OTA*] MUST be supported. WAP Gateway must follow the rules for HTTP proxies (RFC2616) for JAD & JAR MIME-types[*OTA*].
- HTTP authentication (server responses 401 and 407) MUST be supported.
- Set-cookie-headers MUST be passed to the terminal.
- Data of any kind MUST be passed to the terminal, if the terminal's request has included "`Accept: */*`" -header.
- POST-messages from the terminal to provisioning server MUST be supported.

B.5 MIDlet/MIDlet Suite Recommendations

MIDlets SHOULD function correctly even with long connection setup delays and long breaks in connection. Long connection setup delays affect circuit-switched data connections, and long breaks affect GPRS connections.

B.6 References

OTA Over The Air User Initiated Provisioning for Mobile Information Device Profile

MIDP Mobile Information Device Profile Specification 1.0, http://jcp.org/jsr/detail/37.jsp

WAP_JUNE2000 *WAP June2000 Conformance Release* , http://www.wapforum.org/what/technical.htm

WAP_WDPS *WAP Wireless Datagram Protocol Specification*, http://www.wapforum.org/what/technical.htm

B.7 Terms

CSD Circuit Switched Data

GPRS General Packet Radio Service

PPP Point-to-Point Protocol

SMS Short Message Service

USSD Unstructured Supplementary Services Data

WAP Wireless Application Protocol

WSP Wireless Session Protocol