



Java™ Technology for the Wireless Industry

Specification, Version 1.0
Java™ 2 Platform, Micro Edition

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, California 95054
U.S.A.
1-800-555-9SUN or 1-650-960-1300
June, 2003

Java™ Technology for the Wireless Industry Specification ("Specification")

Version: 1.0

Status: FCS

Release: June 26, 2003

Copyright 2003 Sun Microsystems, Inc.

4150 Network Circle, Santa Clara, California 95054, U.S.A

All rights reserved.

NOTICE; LIMITED LICENSE GRANTS

Sun Microsystems, Inc. ("Sun") hereby grants you a fully-paid, non-exclusive, non-transferable, worldwide, limited license (without the right to sublicense), under the Sun's applicable intellectual property rights to view, download, use and reproduce the Specification only for the purpose of internal evaluation, which shall be understood to include developing applications intended to run on an implementation of the Specification provided that such applications do not themselves implement any portion(s) of the Specification.

Sun also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights or patent rights it may have in the Specification to create and/or distribute an Independent Implementation of the Specification that: (i) fully implements the Spec(s) including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and (iii) passes the TCK (including satisfying the requirements of the applicable TCK Users Guide) for such Specification. The foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose.

You need not include limitations (i)-(iii) from the previous paragraph or any other particular "pass through" requirements in any license You grant concerning the use of your Independent Implementation or products derived from it. However, except with respect to implementations of the Specification (and products derived from them) that satisfy limitations (i)-(iii) from the previous paragraph, You may neither: (a) grant or otherwise pass through to your licensees any licenses under Sun's applicable intellectual property rights; nor (b) authorize your licensees to make any claims concerning their implementation's compliance with the Spec in question.

For the purposes of this Agreement: "Independent Implementation" shall mean an implementation of the Specification that neither derives from any of Sun's source code or binary code materials nor, except with an appropriate and separate license from Sun, includes any of Sun's source code or binary code materials; and "Licensor Name Space" shall mean the public class or interface declarations whose names begin with "java", "javax", "com.sun" or their equivalents in any subsequent naming convention adopted by Sun through the Java Community Process, or any recognized successors or replacements thereof.

This Agreement will terminate immediately without notice from Sun if you fail to comply with any material provision of or act outside the scope of the licenses granted above.

TRADEMARKS

No right, title, or interest in or to any trademarks, service marks, or trade names of Sun or Sun's licensors is granted hereunder. Sun, Sun Microsystems, the Sun logo, Java, J2ME, and the Java Coffee Cup logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

DISCLAIMER OF WARRANTIES

THE SPECIFICATION IS PROVIDED "AS IS". SUN MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS. This document does not represent any commitment to release or implement any portion of the Specification in any product.

THE SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SUN MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.



Please
Recycle



Adobe PostScript

LIMITATION OF LIABILITY

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF SUN AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend Sun and its licensors from any claims arising or resulting from: (i) your use of the Specification; (ii) the use or distribution of your Java application, applet and/or clean room implementation; and/or (iii) any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

RESTRICTED RIGHTS LEGEND

U.S. Government: If this Specification is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Specification and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

REPORT

You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your use of the Specification ("Feedback"). To the extent that you provide Sun with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Sun a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the Specification and future versions, implementations, and test suites thereof.

(LFI#132445/Form ID#011801)



Please
Recycle



Adobe PostScript



Please
Recycle



Adobe PostScript

Contents

Preface ix

1. Introduction and Background 1

- 1.1 JSR-185 Expert Group 1
- 1.2 Goals 2
- 1.3 Requirements 2
 - 1.3.1 Mandatory Specifications 3
 - 1.3.2 Conditionally Required Specifications 4
 - 1.3.3 Minimum Configuration 4
 - 1.3.4 Other Specifications 5
- 1.4 Approach to Clarifications 5

2. JTWI Architecture Specification 7

- 2.1 Wireless Technology Architecture 7
 - 2.1.1 Mobile Phone Software Components 7
 - 2.1.2 General Guidelines for Compatibility 8
 - 2.1.3 Versions of Profiles and Optional Packages 8
 - 2.1.4 Optional Functionality within APIs 9
 - 2.1.5 Conformance Testing and Assurance 9
 - 2.1.6 Recommended Resources 10
 - 2.1.7 Standard-Size Application 10
 - 2.1.8 Pre Allocation of Storage for MIDlet Suites 11
 - 2.1.9 Growth of RMS Storage Beyond the Reserved Minimum 12

3. Connected Limited Device Configuration 1.0 (JSR-030)	13
3.1 CLDC 1.0 Requirements	13
3.2 Minimum Application Thread Count	14
3.3 Minimum Clock Resolution	14
3.4 Custom Time Zone IDs	15
3.5 Names for Encodings	16
3.6 Character Properties	17
3.7 Unicode Version	18
4. Mobile Information Device Profile 2.0 (JSR-118)	19
4.1 MIDP 2.0 Compatibility Requirements	19
4.2 Record Store Minimum	20
4.3 HTTP Support for Media Content	20
4.4 JPEG for Image Objects	21
4.5 Timer Resolution	22
4.6 Minimum Number of Timers	22
4.7 Bitmap Minimums	23
4.8 TextField and TextBox and Phone Book Coupling	24
4.9 Supported Characters in TextField and TextBox	25
4.10 Supported Characters in EMAILADDR and URL Fields	27
4.11 Push Registry Alarm Events	27
4.12 Identification of JTWI via System Property	28
5. Wireless Messaging API 1.1 (JSR-120)	29
5.1 WMA Requirements	29
5.2 Support for SMS in GSM Devices	29
5.3 Cell Broadcast in GSM Devices	30
5.4 SMS Push	31
6. Mobile Media API 1.1 (JSR-135)	33
6.1 MMAPI Requirements	33
6.2 HTTP 1.1 Protocol	33
6.3 MIDI Feature Set	34

6.4	Controls for MIDI Feature Set	35
6.5	JPEG Encoding in Video Snapshots	35
6.6	Tone Sequence File Format	36
7.	Security Policy for GSM/UMTS Compliant Devices	37
7.1	Goals, Scope and Requirements	37
7.2	Protection Domains	38
7.2.1	Untrusted Domain	38
7.2.2	Other Protection Domains	38
7.3	Security Settings for MIDlet Suites	39
7.3.1	Permission Function Groups	39
7.3.2	Requirements on Restricted APIs	41
7.3.3	Presenting Function Group Information	41
7.3.4	User Prompts and Notifications	42
7.4	MIDP 2.0 API Untrusted Policy	42
7.5	Wireless Messaging Untrusted Policy	44
7.6	Mobile Media API Untrusted Policy	45

Preface

This document is the Java™ Technology for the Wireless Industry (JTWI) specification, defined by the Java Community ProcessSM (JCP) expert group JSR-185. The document specifies a set of services to develop highly portable, interoperable mobile phone applications.

This section is a guide to this document, and it provides information about the conventions used in this document.

This document and all associated documents are subject to the terms of the JCP agreements (i.e. JSPA and/or IEPA).

Definitions

This document uses definitions based upon those specified in RFC 2119 (See <http://www.ietf.org>).

Term	Definition
must	The associated definition is an absolute requirement of this specification.
must not	The definition is an absolute prohibition of this specification.
should	Indicates a recommended practice. There may exist valid reasons in particular circumstances to ignore this recommendation, but the full implications must be understood and carefully weighed before choosing a different course.
should not	Indicates a non-recommended practice. There may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
may	Indicates that an item is truly optional.

References

ID	Document
1	<i>ISO/IEC JPEG</i> <i>ITU-T Recommendation T.81: "Information technology; Digital compression and coding of continuous-tone still images: Requirements and guidelines" 09/92.</i> http://www.w3.org/Graphics/JPEG/itu-t81
2	<i>JFIF</i> <i>"JPEG File Interchange Format", Version 1.02, September 1, 1992</i> http://www.jpeg.org/public/jfif.pdf
3	<i>CLDC 1.0</i> <i>"Connected, Limited Device Configuration 1.0", version 1.0, May 19, 2000</i> http://www.jcp.org/en/jsr/detail?id=30
4	<i>CLDC 1.1</i> <i>"Connected, Limited Device Configuration 1.1", version 1.1, Mar 20, 2002</i> http://www.jcp.org/en/jsr/detail?id=139
5	<i>MIDP 2.0</i> <i>"Mobile Information Device Profile 2.0", version 2.0, Nov. 20, 2002</i> http://www.jcp.org/en/jsr/detail?id=118
6	<i>WMA 1.1</i> <i>"Wireless Messaging API 1.1", version 1.1, [date TBD 2003]</i> http://www.jcp.org/en/jsr/detail?id=120
7	<i>MMAPI 1.1</i> <i>"Mobile Media API 1.1", version 1.1, [date TBD 2003]</i> http://www.jcp.org/en/jsr/detail?id=135
8	<i>"7-Bit ASCII"</i> <i>7-Bit American National Standard Code for Information Interchange — ANSI INCITS 4-1986 (R2002)</i> http://webstore.ansi.org/ansidocstore/product.asp?sku=ANSI+INCITS+4%2D1986+%28R2002%29
9	<i>"ISO 8859-1"</i> <i>ISO/IEC 8859-1:1998 - 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1. International Organization for Standardization, 1998</i> http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=28245&ICS1=35&ICS2=40&ICS3=

ID	Document
10	<p><i>ISO 646</i> <i>ISO/IEC 646:1991 - Information technology — ISO 7-bit coded character set for information interchange. International Organization for Standardization, 1991.</i> http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=4777&ICS1=35&ICS2=40&ICS3=</p>
11	<p><i>Unicode Reference</i> <i>The Unicode Standard, Version 3.0. Addison-Wesley, 2000</i> http://www.unicode.org/unicode/standard/standard.html</p>
12	<p><i>IANA Official Names for Character Sets</i> http://www.iana.org/assignments/character-sets</p>
13	<p><i>HTTP 1.1 Specification</i> http://www.ietf.org/rfc/rfc2616.txt</p>
14	<p><i>PNG (Portable Network Graphics) Specification</i> http://www.w3.org/TR/REC-png</p>

Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of package and class names	<code>java.lang.String</code> <code>java.util.HashMap.get()</code>
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be superuser to do this.

Feedback

We are interested in improving our specification and welcome your comments and suggestions. You can e-mail your comments to:

jtwi-comments@sun.com

Revision History

Date	Version	Description
23-May-2003	Draft 0.12	Proposed Final Draft
2-June-2003	Draft 0.13	Version 1.0

Introduction and Background

This document is the Java™ Technology for the Wireless Industry (JTWI) specification, defined by the Java Community ProcessSM expert group JSR-185.

The key goal of the JTWI specification is to minimize API fragmentation in the mobile phone device market, and to deliver a predictable, clear specification for device manufacturers, operators, and application developers. Devices, by meeting this specification benefit from a broad range of compatible applications. Software developers, by developing within these constraints, benefit by having a broad range of devices which support their applications.

1.1 JSR-185 Expert Group

The JTWI specification is the result of an industry collaboration effort that consists of a wide range of companies from all over the world. The following companies are members of the JSR-185 expert group that has defined the JTWI specification:

- 4thPass
- Aplix Corporation
- JAMDAT Mobile Inc.
- Matsushita
- Motorola
- Nokia
- NTT DoCoMo, Inc.
- Orange
- Research In Motion, LTD
- Samsung Electronics Corporation
- Siemens AG
- Sony Ericsson Mobile Communications AB
- Sprint PCS
- Sun Microsystems, Inc.
- Symbian, Ltd.
- T-Mobile International AG
- Telefónica Móviles España
- Vodafone Group PLC

Additionally, there are a number of observer members within the expert group.

1.2 Goals

The goal of the JTWI specification is to improve the compatibility, interoperability and completeness of J2ME technology implementations in mobile phones. The JTWI specification minimizes API fragmentation and raises the bar of functionality for high-volume devices thereby broadening the already substantial base of applications that have been developed for mobile phones.

This specification achieves these goals in three ways:

- First, it specifies a common set of APIs by requiring the use of a number of *component specifications* such as the MIDP Specification version 2.0 (MIDP 2.0). In some cases, these specifications provide uniform access to services that were once only available through device specific APIs, or not at all.
- Second, it mandates elements of the component specifications, thereby eliminating optional features and improving compatibility.
- Third, some elements of the component specifications have been clarified within this specification.

The specification leads of the component specifications (such as MIDP 2.0 or WMA 1.1) have been proactive participants in the JSR-185 expert group and the development of this specification. They have been active in proposing clarifications included in this specification, and they have also taken responsibility for additional issues that will be resolved separately in maintenance release of the component specifications.

1.3 Requirements

This specification defines three categories of specifications: *Mandatory* and *Conditionally Required* and *Minimum Configuration*.

Summary of JTWI Specification

Mandatory Specifications:

- MIDP 2.0 (JSR-118)
- WMA 1.1 (JSR-120)

Conditionally Required Specification:

- MMAPI 1.1 (JSR-135)

Minimum Configuration on which JTWI is built:

- CLDC 1.0 (JSR-30)

1.3.1 Mandatory Specifications

For a device to be JTWI-compliant, it must implement each of the mandatory specifications, per the requirements outlined in [Section 2.1.3 “Versions of Profiles and Optional Packages”](#).

MIDP 2.0 (JSR-118)

Rationale

MIDP 2.0 offers a carefully selected set of additional APIs beyond what exists in MIDP 1.0. These APIs address a variety of issues for MIDP 1.0, including:

- Improved gaming services
- Audio extensions
- Security extensions

In addition, MIDP 2.0 substantially clarifies the MIDP 1.0 specification, improving interoperability significantly. Some of the added APIs are present to make higher performance applications possible. The net impact in ROM footprint is no more than 200 KB. The footprint increase will be smaller if an existing native implementation can be leveraged for the new elements of MIDP 2.0.

During the JTWI specification process, the optional functions defined by MIDP 2.0 have been reviewed, and some optional functions have been made mandatory for use with JTWI.

WMA 1.1 (JSR-120)

Rationale

Wireless messaging is a key service that can be used by games, business applications, and commerce applications. Wireless messaging provides a simple and low-cost mechanism to enable application communications.

WMA implementations need to use some underlying messaging technology, but the JTWI specification does not mandate any specific technology to be used.

1.3.2 Conditionally Required Specifications

Specifications in this category have one or more conditions identified in this document that will mandate their inclusion in a compliant implementation. For a specification in this category, there are additional requirements, such as mandating optional elements that are required if the specification is included in a compliant device.

MMAPI 1.1 (JSR-135)

Rationale

MMAPI defines media and user interface features that enable the development of more compelling games and new types of content.

The JTWI specification mandates specific media types that must be supported so that content providers and application developers can depend on a known set of content types. The JTWI specification has separate requirements for devices which support either audio-only or both audio and video capabilities.

The MMAPI specification is required by the JTWI specification under conditions that are detailed in the MMAPI section of this document.

1.3.3 Minimum Configuration

JTWI is built upon a J2ME *configuration*. The *configuration* provides the virtual machine (VM) and the basic APIs of the application environment.

CLDC 1.0 (JSR-30)

Rationale

JTWI is designed to be implemented on top of CLDC 1.0, which provides basic APIs and the VM itself. Other configurations which provide compatible APIs, including CLDC 1.1 are acceptable alternatives.

1.3.4 Other Specifications

The JTWI specification does not explicitly exclude any specification that has been publicly released through the Java Community Process, except those that have been superseded by a specification mandated by the JTWI specification. Compliant devices may also include other APIs as deemed necessary by the device supplier.

1.4 Approach to Clarifications

The remaining chapters of this document describe the specific requirements of optional elements within each specification and also include other compatibility requirements. JTWI does not define any new APIs.

As part of its goal to reduce fragmentation and improve interoperability, JTWI defines some requirements. In the process of developing the requirements for JTWI, a set of desired clarifications were collected. These clarifications were categorized, and those which fit within the scope of JTWI were used to create its requirements.

An overriding consideration for the inclusion of any single requirement was its impact on compatibility and interoperability. The categorization below also affected the decisions to mandate the requirements of the JTWI specification. Other factors, such as practicality of implementation were also considered.

The clarifications fall into this set of categories:

1. *Specification clarifications*

Many of the clarifications attempt to provide an improved description of the intent of the component specification. Clarifications of this type have been coordinated with the specification leads of the component specifications to ensure consistency and coherency of the documentation. These items are a high priority for the expert group, as they introduce the highest risk of unrecoverable incompatibility among devices.

2. *Optional elements*

Many of the component specifications have optional elements. In many cases, these optional elements exist because of the broad range of devices that a particular specification might target.

In some cases, JTWI strengthens these elements by making them mandatory, provided that this is consistent with the kind of device that JTWI is targeting (that is, mobile phones).

3. Items unspecified within the component specification

JTWI does not try to introduce new specifications that would potentially conflict with a future version of a component specification. However, some items that were intentionally left unspecified in the component specification (such as resource formats and media content types in MMAPI) have been integrated into the JTWI specification.

4. Resource and boundary constraints

Clarifications of this nature are those where the component specification defines a service, feature or capability but does not define boundaries or limits for it.

JTWI specifies requirements for some resource boundary constraints when it is believed that the requirement will help improve compatibility. In addition, these requirements act as a guide to MIDlet developers as to the limits of JTWI-compliant mobile phones.

For all of the requirements of JTWI, the intent is to **not** supersede or override the specification of its component specifications. Clarifications that exhibited this condition, but were still seen as critical to improving compatibility have been forwarded to the specification lead of the affected JSR with an appropriate recommendation.

In addition, all of the items that were agreed to be desirable have been passed to the specification leads of the affected specification, regardless of their appearance within the JTWI specification, so as to minimize the risk of collisions in future releases.

JTWI Architecture Specification

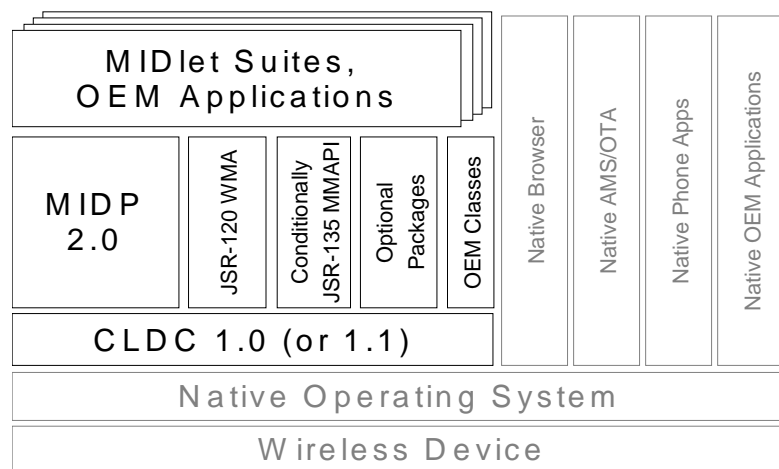
2.1 Wireless Technology Architecture

The diagrams below illustrate the relationships of the software and hardware within the Java runtime environment with the other components on the mobile phone, and the relationship of the wireless device with the wireless network and servers.

2.1.1 Mobile Phone Software Components

FIGURE 1 shows the relationship of the JTWI components to other typical components in the software stack.

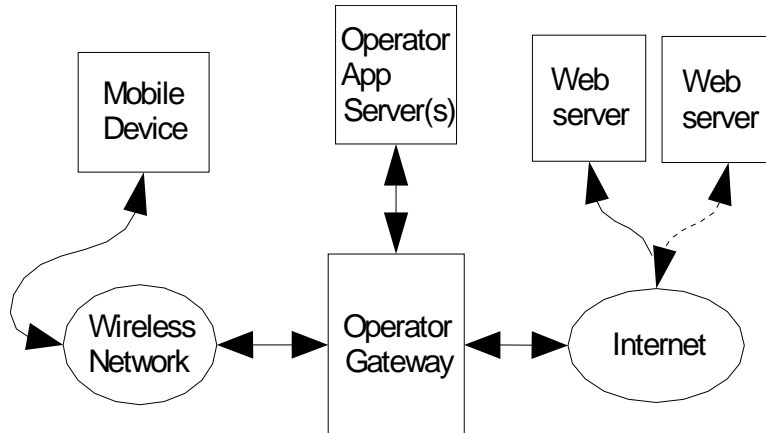
FIGURE 1 Components within a mobile phone



Network Architecture

The mobile phone is the client component of the service delivery platform. [FIGURE 2](#) shows the relationships between the client, the wireless network, the internet and the various servers that provide the total service delivery platform.

FIGURE 2 Components of a typical service platform



2.1.2 General Guidelines for Compatibility

This section defines guidelines for composing a configuration, a profile and optional packages to achieve a coherent and architecturally consistent implementation.

2.1.3 Versions of Profiles and Optional Packages

In general, all versions of a configuration, profile or optional package are upward compatible with previous versions. Each implementation of JTWI:

- **must** implement at least the lowest version of the API as identified by this specification
- **may** implement a higher version or superset of the same API

For example, CLDC 1.0 (JSR-30) and CLDC 1.1 (JSR-139) can both be used as the configuration on top of which JTWI is implemented. CLDC 1.0 is the minimum version allowed but CLDC 1.1, as a compatible configuration, can also be used. Similarly, the Audio Building Block defined for MIDP 2.0 is a strict subset of the full MMAPI Mobile Media API and the full MMAPI specification can be used instead of the Audio Building Block.

In all cases, the applicable Technology Compatibility Kit (TCK) tests apply and **must** be run to verify conformance.

2.1.4 Optional Functionality within APIs

Many specifications have by necessity included some optional specifications. These can be conditional, implementation-optional or open-ended. From the perspective of developers, system integrators, manufacturers and operators, each option increases the difficulty of understanding what is available on a particular device. Generally, JTWI seeks to reduce the number of options. To support that goal, JTWI may promote particular functions from **may**, **should** or optional status to **must** be implemented (mandatory). JTWI may also apply additional requirements to the configuration or the other components.

In the conditional case, there may be a function that is only required to be implemented under particular conditions. For example, MIDP 2.0 requires support for the WAV media format if any sampled sound format is included.

In the implementation optional case, the interface to and behavior of the function is specified but there is a well-defined exception if the implementation is not present. If the implementation is present, then it **must** be complete and faithfully implement the specification. The TCK **must** deal with this option and enable the correct tests as appropriate. JTWI would make the implementation mandatory and require corresponding TCK tests.

In the open-ended case, the API is designed to be agnostic to a particular implementation choice. However, for an application to work correctly and portably, additional specification is needed. For example, MMAPAPI and the MIDP 2.0 Audio Building Block do not require specific media formats and can operate correctly only if the media presented to the API matches the media type supported by the implementation. If there is a mismatch, then the MIDlet is non-portable.

2.1.5 Conformance Testing and Assurance

The role of conformance testing for JTWI is a critical component of ensuring application portability and interoperability. Each specification that is mandated by this specification **must** have a TCK that is sufficiently rigorous to ensure that the various implementations of the API operate as expected by MIDlet suites and that other functional behavior of the API works as specified.

A JTWI-compliant implementation must pass the TCKs for all of the implemented JCP specifications.

2.1.6 Recommended Resources

The JTWI specification places various requirements on minimum capacities to process programs and information, for example, specifying a minimum size of a MIDlet suite that must be supported.

This specification recommends several device minimums to allow application developers to choose appropriate device targets, and device manufacturers to target appropriate systems. These recommendations are summarized in [TABLE 1](#). These device recommendations are in addition to the minimum requirements specified in the MIDP 2.0 specification.

TABLE 1 Recommended Resource Minimums for Various Features

Feature	Recommendation
Screen size as returned by full screen mode <code>Canvas.getHeight()</code> and <code>Canvas.getWidth()</code>	125x125 pixels
Color depth as returned by <code>Display.numColors()</code>	4096 colors (12 bits)
Pixel aspect ratio	1:1
Volatile memory for Java runtime (for example, the Java Heap)	256 KB

Developers are encouraged to ensure applications can run within these bounds and the bounds defined within the MIDP 2.0 specification.

2.1.7 Standard-Size Application

This specification defines the concept of a standard-size application. The value proposition for developers is that applications written to operate within the standard size works correctly on all devices compliant with this specification and can reach the greatest number of customers. Many devices are likely to support larger applications up to limits determined by the amount of memory on the device, less the amount due to other installed applications or data that consumes the memory, or by limits set by the operator to meet quality-of-service requirements.

When targeting maximum portability between different devices the developer should ensure that the MIDlet suite and its MIDlets do not attempt to use more resources than listed below. While applications will need to handle out of memory conditions, operation up to these limits should be supported. With the standard-size application definition it is possible to determine how many standard-size applications can be installed on a device.

TABLE 2 Standard-size Application Definition

Feature	Size Limit	Actual Size
JAR Size	64 KB	The actual size of the MIDlet suite JAR is used
Application Descriptor Size	5 KB	The actual size of the application descriptor (JAD) is used
RMS Data Size	30 KB	The actual size is set from the MIDlet-Data-Size attribute. MIDlet-Data-Size should be the minimum amount of RMS that the application can work with.

2.1.8 Pre Allocation of Storage for MIDlet Suites

To ensure the application has the resources it needs to run consistently, the device implementation **must** meet the storage requirements of the MIDlet suite listed in [TABLE 2](#). If the storage requirements cannot be met, the installation **must** fail and the MIDlet suite JAR should not be downloaded. The implementation **must** be able to download and install a MIDlet suite of the indicated JAR size and application descriptor size. The RMS Data Size **must** be reserved for the application at all times so it may be started at any time and run consistently as long as it stays within the actual limits for RMS data size.

A conforming implementation **must** meet the resource requirements as follows:

TABLE 3 MIDlet Pre-allocation Requirements

Feature	Compatibility Criteria
JAR Size	Successful installation of a JAR consisting of a simple MIDlet and resource files to fill the JAR to the JAR size limit.
Application Descriptor Size	Successful installation of a MIDlet suite with an application descriptor of the Application Descriptor Size limit.
RMS Data Size	Successful installation of a MIDlet suite requesting the RMS Data Size limit for MIDlet-Data-Size and running a MIDlet creating a single RecordStore and reporting that the RecordStore.getSizeAvailable() method returns a value greater than or equal to the MIDlet-Data-Size attribute.

2.1.9 Growth of RMS Storage Beyond the Reserved Minimum

The MIDlet suite may attempt to store more information than the amount reserved by the `MIDlet-Data-Size` attribute. In this case the MIDlet suite is responsible for correct operation even if additional space is not available. When the amount of information stored is reduced below the level set by `MIDlet-Data-Size`, the unused storage up to `MIDlet-Data-Size` remains reserved for the MIDlet suite.

Application developers should carefully choose what size to specify for the `MIDlet-Data-Size` attribute. This attribute specifies the minimum amount of RMS space that the MIDlet suite can work with and how much RMS space will be permanently reserved for the MIDlet suite at installation time. Since nothing prevents applications from using more space later, it is advisable to specify the `MIDlet-Data-Size` to be as small as possible. Having a smaller `MIDlet-Data-Size` will increase the likelihood that there is enough space in the end user's device to install the MIDlet suite.

Connected Limited Device Configuration 1.0 (JSR-030)

The JTWI specification uses CLDC 1.0 as the minimum required J2ME *configuration*. A *configuration* of the J2ME platform specifies a subset of the Java programming language features and Java virtual machine features, as well as the core platform libraries, to support a wide range of consumer products.

CLDC 1.1 is a strict super set of CLDC 1.0 and **may** be used instead of CLDC 1.0 in JTWI implementations. Other configurations developed through the JCP are not within the scope of this specification but are not excluded. For all configurations, the applicable CLDC requirements identified below are required.

If floating point capabilities are exposed to Java applications, CLDC 1.1 or another compatible J2ME configuration **must** be implemented.

3.1 CLDC 1.0 Requirements

JTWI **must** be implemented on top of CLDC 1.0 or a compatible J2ME configuration and the corresponding TCK **must** be passed.

3.2 Minimum Application Thread Count

Applicable Document, Section, Classes and Methods

CLDC 1.0 and *CLDC 1.1*

```
class java.lang.Thread
```

Requirement Text

A compliant implementation **must** allow a MIDlet suite to create a minimum of 10 simultaneously running threads.

Justification/Notes

Although many implementations can support more threads, MIDlet developers are encouraged to stay within these bounds. This requirement is not intended to require implementations guarantee at all times that 10 threads be possible, but that implementations not artificially limit thread creation to less than 10 threads. Application developers must still manage resource usage within the physical constraints of the device.

3.3 Minimum Clock Resolution

Applicable Document, Section, Classes and Methods

CLDC 1.0 and *CLDC 1.1*.

```
class java.lang.System:  
    public long currentTimeMillis()
```

Requirement Text

In a compliant device, the method `java.lang.System.currentTimeMillis()` **must** record the elapsed time in increments not to exceed 40 milliseconds. Various factors, such as garbage collection, affect the ability to achieve this requirement. Because of this, at least 80% of test attempts **must** meet the time elapsed requirement to achieve acceptable conformance.

Justification/Notes

A compliant implementation **may** support a smaller increment.

3.4 Custom Time Zone IDs

Applicable Document, Section, Classes and Methods

[CLDC 1.0](#) and [CLDC 1.1](#).

```
class java.util.TimeZone:  
    public String getID()  
    public static TimeZone getTimeZone(String ID)
```

Requirement Text

A compliant implementation **must** permit the use of custom time zones which adhere to the following time zone format:

General time zone: For time zones representing a GMT offset value, the following syntax is used:

- *CustomID:*

GMT Sign Hours : Minutes

GMT Sign Hours Minutes

GMT Sign Hours

- *Sign:* one of:

+ -

- *Hours:*

Digit

Digit Digit

- *Minutes:*

Digit Digit

- *Digit:* one of:

0 1 2 3 4 5 6 7 8 9

Hours **must** be between 0 and 23. *Minutes* **must** be between 00 and 59. For example, “GMT+10” and “GMT+0010” mean ten hours and ten minutes ahead of GMT, respectively. The format is locale independent and *digits* **must** be taken from the

Basic Latin block of the Unicode standard. No daylight saving time transition schedule can be specified with a custom time zone ID. If the specified string doesn't match the syntax, "GMT" is used.

When creating a `TimeZone`, the specified custom time zone ID is normalized in the following syntax:

NormalizedCustomID:

GMT Sign TwoDigitHours : Minutes

Sign: one of

+ -

TwoDigitHours:

Digit Digit

Minutes:

Digit Digit

Digit: one of

0 1 2 3 4 5 6 7 8 9

For example, `TimeZone.getTimeZone("GMT-8").getID()` returns `GMT-08:00`.

Justification/Notes

CLDC 1.0 and CLDC 1.1 mandate only the time zone ID "GMT" for the `TimeZone` class. Mandating support for the custom time zone format provides consistent time zone ID support across implementations.

3.5 Names for Encodings

Applicable Document, Section, Classes and Methods

[CLDC 1.0](#) in Section 6.2.8 and [CLDC 1.1](#), in Section 6.2.9.

```
class java.io.InputStreamReader
class java.io.OutputStreamReader
class java.lang.String:
    public String(byte[] bytes, int off, int len, String enc)
    public String(byte[] bytes, String enc)
    public byte[] getBytes(String enc)
```

Requirement Text

Implementations **must** support at least the preferred MIME name as defined by IANA (<http://www.iana.org/assignments/character-sets>) for the supported character encodings. For example, for [ISO 646](#), the name “US-ASCII” **must** be supported. If no preferred name has been defined, then the registered name **must** be used, for example, “UTF-16”.

Justification/Notes

Application developers are encouraged to use the preferred names in order to ensure portability. Although devices are permitted to use the other names, this is intended only for backwards compatibility.

3.6 Character Properties

Applicable Document, Section, Classes and Methods

[CLDC 1.0](#) and [CLDC 1.1](#), in Section 6.2.1.

```
class java.lang.String:
    public int compareTo(String s)
    public boolean regionMatches(boolean ignoreCase, int toffset,
        String other, int ooffset, int len)

class java.lang.Character:
    public boolean equals(Object o)
    public static boolean isLowerCase(char ch)
    public static boolean isUpperCase(char ch)
    public static boolean isDigit(char ch)
    public static char toLowerCase(char ch)
    public static char toUpperCase(char ch)
    public static int digit(char ch, int radix)
```

Requirement Text

Compliant implementations **must** provide support for character properties and case conversions for characters in the “Basic Latin” and “Latin-1 Supplement” blocks of Unicode 3.0. Other Unicode character blocks may be supported as necessary.

Justification/Notes

The CLDC 1.0 specification permits more relaxed case-conversion. However, applications which target many international markets will be less satisfactory for their users if case conversion is incomplete.

3.7 Unicode Version

Applicable Document, Section, Classes and Methods

[CLDC 1.0](#) in section 6.2.8 and [CLDC 1.1](#), in Section 6.2.9.

```
class java.lang.String
class java.lang.StringBuffer
class java.lang.Character
```

Requirement Text

A compliant implementation **must** support Unicode characters. Character information is based on the Unicode Standard, version 3.0. However, since the full character tables required for Unicode support can be excessively large for devices with tight memory budgets, by default the character property and case conversion facilities in CLDC assume the presence of ISO Latin-1 range of characters only. See [Section 3.6 “Character Properties”](#) for more details.

Justification/Notes

This is addressed in CLDC 1.1; this requirement is included to ensure that both CLDC 1.0- and CLDC 1.1-based implementations are consistent.

Mobile Information Device Profile 2.0 (JSR-118)

MIDP 2.0 provides the library support for user interface, persistent storage, networking, security and push functions. MIDP 2.0 contains a number of optional functions, some of which must be implemented as described below. The JTWI requirements for the MIDP 2.0 implementation are described below.

4.1 MIDP 2.0 Compatibility Requirements

The goal of this section is to reduce the incompatibilities and variations between implementations that developers must overcome by decreasing the number of optional functions allowed by MIDP 2.0.

MIDP 2.0 or a subsequent compatible version **must** be implemented and the corresponding TCK **must** be passed by JTWI-compliant implementations.

An implementation of JTWI **may** implement any optional function of MIDP, but some optional elements of MIDP 2.0 are mandated by JTWI.

4.2 Record Store Minimum

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 14.

```
class javax.microedition.rms.RecordStore
```

Requirement Text

A compliant implementation **must** permit a MIDlet suite to create at least 5 independent RecordStores.

Justification/Notes

Most implementations will not have artificial limits on the number of RecordStores; if such a limit exists, it should be at least 5. This requirement does not intend to mandate that memory be reserved for these RecordStores, but it **must** be possible to create the RecordStores if the required memory is available. MIDlets are encouraged to use this resource carefully, and consider minimizing the total number of RecordStores required for the application.

4.3 HTTP Support for Media Content

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 10.

```
class javax.microedition.media.Manager:  
    public static String getSupportedProtocols()  
    public static Player createPlayer()  
    public static String getSupportedContentTypes()
```

Requirement Text

Compliant devices **must** provide support for HTTP 1.1 for all supported media types. HTTP 1.1 conformance **must** match the MIDP 2.0 specification. See package `javax.microedition.io` for specific requirements.

Justification/Notes

A predictable method for acquiring content is necessary to maximize the value of the application environment, and to simplify the development effort required for client/server applications.

4.4 JPEG for Image Objects

Applicable Document, Section, Classes and Methods

[MIDP 2.0](#), Chapter 1, Section PNG Image Format and Chapter 8.

```
class javax.microedition.lcdui.Image:
    public static Image createImage(byte[] imageData, int
        imageOffset, int imageLength)
    public static Image createImage(Image image, int x, int y, int
        width, int height, int transform)
    public static Image createImage(java.io.InputStream stream)
    public static Image createImage(String name)
```

Requirement Text

[ISO/IEC JPEG](#) together with [JFIF](#) **must** be supported by a compliant device. The support for [ISO/IEC JPEG](#) only applies to baseline DCT, non-differential, Huffman coding, as defined in table B.1, symbol 'SOF0' in [1]. This support extends to the class `javax.microedition.lcdui.Image`, including the methods outlined above.

This mandate is voided in the event that the JPEG image format becomes encumbered with licensing requirements.

Justification/Notes

The only mandatory image format in MIDP 2.0 is PNG. PNG is a lossless format and may consume more space than necessary for images.

4.5 Timer Resolution

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 6.

```
class java.util.Timer:
    public void schedule(TimerTask task, Date time)
    public void schedule(TimerTask task, Date firstTime,
        long period)
    public void schedule(TimerTask task, long delay)
    public void schedule(TimerTask task, long delay, long period)
```

Requirement Text

A compliant implementation **must** permit an application to specify the values for the `firstTime`, `delay`, and `period` parameters of `java.util.Timer.schedule()` methods with a distinguishable resolution of no more than 40 ms. Various factors, such as garbage collection, affect the ability to achieve this requirement. Because of this, at least 80% of test attempts **must** meet the schedule resolution requirement to achieve acceptable conformance.

Justification/Notes

Predictability is crucial to application interoperability. This mandate is provided also to set expectations for application developers.

4.6 Minimum Number of Timers

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 6.

```
class java.util.Timer:
    public void schedule(TimerTask task, Date time)
    public void schedule(TimerTask task, Date firstTime,
        long period)
    public void schedule(TimerTask task, long delay)
    public void schedule(TimerTask task, long delay, long period)
```

Requirement Text

A compliant implementation must allow a MIDlet suite to create a minimum of 5 simultaneously running Timers. This requirement is independent of the minimum requirement specified in [Section 3.2 “Minimum Application Thread Count.”](#)

Justification/Notes

Predictability is crucial to application interoperability. This mandate is also provided to set expectations for application developers. This requirement is not intended to require implementations guarantee at all times that 5 timers be possible, but that implementations not artificially limit timer creation to less than 5 timers. Application developers must still manage resource usage within the physical constraints of the device.

4.7 Bitmap Minimums

Applicable Document, Section, Classes and Methods

[MIDP 2.0](#), Chapter 8, Section PNG Image Format.

```
class javax.microedition.lcdui.Image:
    public static Image createImage(byte[] imageData, int
        imageOffset, int imageLength)
    public static Image createImage(Image source)
    public static Image createImage(Image image, int x, int y, int
        width, int height, int transform)
    public static Image createImage(java.io.InputStream stream)
    public static Image createImage(String name)
```

Requirement Text

A compliant device **must** support the loading of PNG images with pixel color depths of 1, 2, 4, 8, 16, 24, and 32 bits per pixel per the PNG image format specification. For each of these color depths as well as for JFIF image formats, a compliant implementation **must** support images up to 32768 total pixels.

Justification/Notes

Although devices are not mandated to have color displays, they **must** be able to process color images.

4.8 TextField and TextBox and Phone Book Coupling

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 8.

```
class javax.microedition.lcdui.TextBox  
class javax.microedition.lcdui.TextField
```

Requirement Text

Compliant devices **must** implement a mechanism for selecting a phone number from the device phone book when the user is editing a `TextBox` or `TextField` and the constraint of the `TextBox` or `TextField` is `TextField.PHONENUMBER`. This requirement is voided if the phone book is not accessible.

Justification/Notes

Applications that use wireless messaging are more usable if the user is provided with a simple means of copying data from the phone book. Other applications may also benefit from this mechanism. Note that at no time should the MIDlet be given direct access to the contents of the device phone book itself. This is intended as a convenience for the user to select a number from the stored phone book and insert it into the text field.

4.9 Supported Characters in TextField and TextBox

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 8.

```
class javax.microedition.lcdui.TextBox  
class javax.microedition.lcdui.TextField
```

Requirement Text

TextBox and TextField with input constraint TextField.ANY **must** support inputting the following set of characters:

TABLE 4 Characters Supported by TextBox and TextField

Glyph	Unicode Value	Description
	U+000A	line feed
	U+0020	space
!	U+0021	exclamation mark
“	U+0022	quotation mark
#	U+0023	number sign
\$	U+0024	dollar sign
%	U+0025	percent
&	U+0026	ampersand
‘	U+0027	apostrophe
(U+0028	left parenthesis
)	U+0029	right parenthesis
*	U+002A	asterisk
+	U+002B	plus
,	U+002C	comma
-	U+002D	hyphen-minus
.	U+002E	period

TABLE 4 Characters Supported by `TextBox` and `TextField`

Glyph	Unicode Value	Description
/	U+002F	slash
0-9	U+0030 - U+0039	digit 0 through 9
:	U+003A	colon
;	U+003B	semicolon
<	U+003C	less-than sign
=	U+003D	equals sign
>	U+003E	greater-than sign
?	U+003F	question mark
@	U+0040	commercial at
A-Z	U+0041 - U+005A	Latin capital letter A through Z
[U+005B	left square bracket
\	U+005C	backslash
]	U+005D	right square bracket
^	U+005E	circumflex accent
_	U+005F	spacing underscore
'	U+0060	grave accent
a-z	U+0061 - U+007A	Latin small letter a through z
{	U+007B	left curly bracket
	U+007C	vertical bar
}	U+007D	right curly bracket
~	U+007E	tilde
¡	U+00A1	inverted exclamation
£	U+00A3	pound sign
¤	U+00A4	currency sign
¥	U+00A5	yen sign
§	U+00A7	section mark
¿	U+00BF	inverted question mark
€	U+20AC	euro sign

Justification/Notes

A device may also implement additional input methods that would allow the entry of additional characters from the full Unicode set.

4.10 Supported Characters in EMAILADDR and URL Fields

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 8.

```
class javax.microedition.lcdui.TextBox  
class javax.microedition.lcdui.TextField
```

Requirement Text

Instances of these classes with either of the constraints `TextField.EMAILADDR` and `TextField.URL` **must** allow the same characters to be input as are allowed for input constraint `TextField.ANY`.

Justification/Notes

E-mail address and URLs can contain any characters. However, the current MIDP-2.0 Specification allows implementations to arbitrarily limit what special characters can be input to URLs and e-mail addresses in `TextField` and `TextBox`.

4.11 Push Registry Alarm Events

Applicable Document, Section, Classes and Methods

MIDP 2.0, Chapter 8.

```
class javax.microedition.io.PushRegistry:  
    public static long registerAlarm(String midlet, long time)
```

Requirement Text

A compliant implementation **must** implement alarm-based push registry entries. If no other security mechanism (such as described in [Chapter 7](#)) is present, the `PushRegistry` Alarm function **must not** be allowed without explicit user permission.

Justification/Notes

This provides a practical means for alarm and calendar-based applications.

4.12 Identification of JTWI via System Property

Applicable Document, Section, Classes and Methods

MIDP 2.0, Page 37 “System Properties”

```
class java.lang.System:  
    public final static String getProperty(String key)
```

Requirement Text

To identify a compliant device and the implemented version of this specification the value of the system property `microedition.jtwi.version` **must** be “1.0”

Justification/Notes

Application programs can identify a JTWI-compliant device and determine the version of implemented specification by examining the value of the property.

Wireless Messaging API 1.1 (JSR-120)

WMA defines an API used to send and receive short messages. The API provides access to network-specific short message services such as GSM SMS or CDMA short messaging.

5.1 WMA Requirements

WMA 1.1 or a subsequent compatible version **must** be implemented and the corresponding TCK **must** be passed by JTWI-compliant implementations.

An implementation of JTWI **may** implement any optional function of WMA, but some optional elements of WMA 1.1 are mandated by JTWI.

5.2 Support for SMS in GSM Devices

Applicable Document, Section, Classes and Methods

[WMA 1.1](#), Appendix A.

This mandate has a direct or indirect impact on many of the APIs that are part of the WMA specification.

Requirement Text

GSM/WCDMA (UMTS) phones **must** support the GSM SMS service by using this API as specified in Appendix A of the JSR-120 Wireless Messaging API specification.

Justification/Notes

Device-to-device communication is a critical element of a broad range of mobile applications. The JTWI specification requires this service to enable these new application classes.

5.3 Cell Broadcast in GSM Devices

Applicable Document, Section, Classes and Methods

[WMA 1.1](#), Appendix B.

This mandate has a direct or indirect impact on many of the APIs that are part of the WMA specification.

Requirement Text

If the implementation supports access to GSM Cell Broadcast via Java APIs, it **must** follow the specification found in Appendix B of [Section WMA 1.1](#).

Justification/Notes

Although not every device or network supports Cell Broadcast, uniform access on the devices that do support it is important to improving application portability.

5.4 SMS Push

Applicable Document, Section, Classes and Methods

[WMA 1.1](#), Appendix A. [MIDP 2.0](#), Chapter 3, Section

`javax.microedition.io.PushRegistry`.

`class javax.microedition.io.PushRegistry`

Requirement Text

For GSM/WCDMA (UMTS) phones **must** support MIDP 2.0 Push handling for the SMS protocol as specified in Appendix D of the [WMA 1.1](#). If no other security mechanism (such as described in [Chapter 7](#)) is present, the PushRegistry SMS Push function **must** not be allowed without explicit user permission.

Justification/Notes

SMS Push provides a mechanism for sophisticated server-driven applications. Role playing and other games, as well as various types of notification and event-driven applications, such as travel reservation systems, can benefit from this capability. However, there are security ramifications which devices must consider when implementing this service.

Mobile Media API 1.1 (JSR-135)

The Mobile Media API enables an application to playback and record various time-based media. Support for specific media formats and network protocols is specified below.

6.1 MMAPI Requirements

MMAPI **may** be implemented by JTWI implementations. If MMAPI is implemented, then version 1.1 **must** be the minimum implemented version.

A compliant device **must** implement MMAPI support for media services that are exposed through Java language APIs. Compliant implementations **should** expose support for these media services only through these Java APIs. Certain optional features of the MMAPI specification are mandated for devices compliant with this specification; the remaining optional items remain optional.

6.2 HTTP 1.1 Protocol

Applicable Document, Section, Classes and Methods

[MMAPI 1.1](#), Section "Overview of MMAPI".

```
class javax.microedition.media.Manager:  
    public static String getSupportedProtocols()  
    public static Player createPlayer()  
    public static String getSupportedContentTypes()
```

Requirement Text

HTTP 1.1 **must** be supported for media file download for all supported media formats.

Justification/Notes

MMAPI does not specify any mandatory protocols. It specifically states that protocols must be defined in profiles. In keeping with the general-purpose nature of MMAPI, it leaves the supported set of protocols and content formats up to the implementation. It also does not make any requirements on which content types should work over which protocol.

6.3 MIDI Feature Set

Applicable Document, Section, Classes and Methods

[MMAPI 1.1](#), Section javax.microedition.media.

```
class javax.microedition.media.Manager:  
    public static Player createPlayer()  
    public static String getSupportedContentTypes()
```

Requirement Text

A compliant device **must** implement the MIDI feature set specified in MMAPI (JSR-135). MIDI file playback **must** be supported.

Justification/Notes

Many games are substantially enhanced by suitable audio services. MIDI provides a relatively compact representation, while providing substantial capabilities at the same time.

6.4 Controls for MIDI Feature Set

Applicable Document, Section, Classes and Methods

MMAPI 1.1, Section `javax.microedition.media`

```
class javax.microedition.media.Player:  
    public Control getControl(String controlType)  
    public Control [] getControls()
```

Requirement Text

Support for `VolumeControl` **must** be implemented.

Justification/Notes

`VolumeControl` is required for controlling the volume of MIDI file playback. It should be noted that the implementation of `MIDIControl` is not mandatory.

6.5 JPEG Encoding in Video Snapshots

Applicable Document, Section, Classes and Methods

MMAPI 1.1, Section `javax.microedition.media`.

```
class javax.microedition.media.Player:  
    public Control getControl(String controlType)  
    public Control [] getControls()
```

Requirement Text

A compliant implementation that supports the video feature set and video image capture **must** support JPEG encoding in Video Snapshots as specified in [Section 4.4 “JPEG for Image Objects.”](#)

Justification/Notes

The PNG image format is not always suitable for photographic applications; JPEG is generally more compact and appropriate when storing photographic images.

6.6 Tone Sequence File Format

Applicable Document, Section, Classes and Methods

MMAPI 1.1, Section `javax.microedition.media`.

```
class javax.microedition.media.Manager:
    public static String []
        getSupportedContentTypes(String protocol)
    public static Player createPlayer(DataSource source)
    public static Player createPlayer(InputStream stream,
        String type)
    public static Player createPlayer(String locator)
```

Requirement Text

Tone sequence file format **must** be supported.

Justification/Notes

Tone Sequences provide an additional simple format for supporting the audio needs of many types of games and other applications.

Security Policy for GSM/UMTS Compliant Devices

The JSR-185 Expert Group recognizes security is an essential element of a complete specification for mobile phones. At the same time, this expert group also acknowledges that many different factors influence the security policy needs of a device.

This portion of this specification is derived from the untrusted portions of the “*Recommended Security Policy for GSM/UMTS Compliant Devices*” (*RP*), an addendum to the [MIDP 2.0](#). Some of the language in the specification has been changed, for several reasons:

- To clarify the required behavior
- To clarify the requirements
- To improve the usability and ease of implementation of the security policy
- To add some elements necessary for network security

For untrusted applications, this specification supersedes the *RP* document. In all other regards, the *RP* is the guideline for device manufacturers to create a complete security policy for JTWI-compliant devices. Future major revisions of this document may incorporate a security policy based on experience gained from the *RP* and evolving understanding of security requirements.

7.1 Goals, Scope and Requirements

MIDP 2.0 defines the framework for authenticating the source of a MIDlet suite and authorizing the MIDlet suite to perform protected functions by granting permissions it may have requested based on the security policy on the device. It also identifies functions that are deemed security vulnerable and defines permissions for those protected functions. Additionally, MIDP 2.0 specifies the common rules for APIs that can be used together with the MIDP but are specified outside the MIDP. The MIDP 2.0 specification does not mandate a single security policy but rather allows the device security policy to be determined by security requirements.

The purpose of this specification is to extend the base MIDlet suite security framework defined in MIDP 2.0 and to define the following areas:

- The required trust model for GSM/UMTS compliant devices
- Capabilities of MIDlets based on permissions defined by MIDP 2.0 and other JCP specifications
- The definition of user permission interaction modes
- Guidelines on user prompts and notifications

GSM/UMTS compliant devices implementing this Security Policy **must** follow the security framework specified in the MIDP 2.0. Non GSM/UMTS devices are encouraged to adopt similar security models.

7.2 Protection Domains

This document specifies requirements for the *untrusted* domain. Other domains are outside the scope of this specification. MIDlets suites that are *untrusted* belong to the untrusted domain. An untrusted MIDlet suite is a MIDlet suite whose origin and integrity cannot be reliably determined.

A protection domain is a way to differentiate between MIDlet suites based on the potentially present digital signature in the MIDlet suite, and to grant or make available to a MIDlet suite a set of permissions. A MIDlet suite cannot belong to more than one protection domain. The representation of a domain and its security policy is implementation specific.

MIDlet suites signed using the MIDP 2.0 signing mechanisms **must not** be installed as untrusted.

7.2.1 Untrusted Domain

A user **must** be informed whenever a new MIDlet suite is installed in the untrusted domain. The notification **must** indicate that the application does not come from a trusted source. The user **must** be able to make an informed decision based on the available information before granting permissions to an application.

For user to make security decisions, appropriate information about the source and trustworthiness of the MIDlet suite **must** be available to the user.

7.2.2 Other Protection Domains

A compliant device **may** implement other protection domains, such as those defined in [MIDP 2.0 Addendum "The Recommended Security Policy for GSM/UMTS Compliant Devices."](#)

7.3 Security Settings for MIDlet Suites

The device implementation **must** prompt the user for permission according to the security policies specified in [TABLE 5](#) through [TABLE 10](#) in the following sections.

[TABLE 5](#) specifies the function groups and the available user permissions for MIDlet suites in the untrusted domain. [TABLE 6](#), [TABLE 7](#) and [TABLE 10](#) specify the mapping of permissions and APIs onto different function groups for each API. [TABLE 8](#) and [TABLE 9](#) specify the effects on the function group.

7.3.1 Permission Function Groups

A device with a small display may not be able to present all permissions to the user in a single configuration settings menu in a user-friendly manner. Therefore the device is not required to present all individual permissions for user confirmation. Rather, a certain higher-level action triggered by the protected function should be brought to the user for acceptance. The high-level functions presented to the user essentially capture and reflect the actions and consequences of the underlying individual permissions. The function groups are as follows:

Network/cost-related groups:

Net Access – contains permissions to any function that results in an active network data connection (for example GSM, GPRS, UMTS, etc.); such functions **must** be mapped to this group

Messaging – contains permissions to any function that allows sending or receiving messages (for example, SMS, MMS, etc.)

Application Auto Invocation – contains permissions to any function that allows a MIDlet suite to be invoked automatically (for example, push, timed MIDlets, etc.)

Local Connectivity – contains permissions to any function that activates a local port for further connection (for example, COMM port, IrDa, Bluetooth, etc.)

User-privacy-related groups:

Multimedia recording – contains permissions to any function that gives a MIDlet suite the ability to capture still images, or to record video or audio clips.

Additional Groups

Whenever new APIs are added to this specification, they **should** be assigned to the appropriate function group. In addition, APIs that are specified elsewhere (that is, in other JCP specifications) but rely on the MIDP security framework, **should** also be assigned to an appropriate function group. If none of the function groups defined in this section is able to capture the new feature and reflect it to the user adequately, then a new function group **must** be defined in this document.

If a new function group is to be added, the following should be taken into consideration: the group to be added **must not** introduce any redundancy to the existing groups; the new group **must** be capable of protecting a wide range of similar features. The latter requirement is to prevent introducing narrowly scoped groups.

The function groups and not the individual permissions should be presented when the user is prompted. Furthermore, the function groups should be presented to the user in the settings of a given MIDlet suite.

Function Group Settings

For each function group, [TABLE 5](#) defines the user permission interaction modes for the untrusted domain. The interaction modes specify how to prompt the user to grant permissions. See [MIDP 2.0](#) page 26 for a description of these interaction modes.

Interaction modes that are not specified in the table **must not** be presented to the user. The *default* is the initial setting until changed by the user.

The names of the function groups are implementation specific but **must** follow the guidelines for the function group names defined in this document as well as the definitions of these groups.

[TABLE 6](#), [TABLE 7](#), and [TABLE 10](#) identify each permission defined in the MIDP 2.0 and other JSRs in the corresponding function group. Each permission **must** occur in only one function group.

TABLE 5 Function Groups and User Settings for the Untrusted Domain

Function Group	Default Setting	Allowed Settings
Net Access	Session	Session, Oneshot, No
Messaging	Oneshot	Oneshot, No
Application Auto Invocation	Session	Session, Oneshot, No
Local Connectivity	Session	Blanket, Session, Oneshot, No
Multimedia Recording	Oneshot	Session, Oneshot, No

7.3.2 Requirements on Restricted APIs

When the user grants permission to a function group, this action effectively grants access to all individual permissions under this function group.

An implementation **must** guarantee that a `SecurityException` is thrown when the caller does not have the appropriate security permissions.

If a MIDlet uses the capabilities defined in MIDP and other APIs, the following rules **must** apply:

- All the external API functions that need to be protected by MIDP 2.0 security framework **must** have permissions defined in those JSRs, and follow the naming rules identified in the MIDP 2.0 Specification, titled “Security for MIDP Applications.”
- The functions that are not deemed security-protected by specification can be accessed explicitly by untrusted MIDlet suites, as per general MIDP security rules.
- If an external API does not define permissions for security-protected functions because the API specification is released earlier than MIDP 2.0, any functions that relate to network access **must** still have the user prompt implemented by the device.
- A device cannot access the network without appropriate user notification.
- All licensee open classes **must** adhere to the permission framework as defined in this document.

7.3.3 Presenting Function Group Information

Untrusted MIDlet suites may request permissions using the attributes in the application descriptor or JAR manifest. The requested *user permissions* are presented using function groups. The presentation is implementation-specific, but the following rules **must** be followed.

- The user **must** be able to change each function group setting to any allowed setting for the function group listed in [TABLE 5](#).
- Each function group has allowed settings that apply to each individual permission within the function group. [TABLE 6](#), [TABLE 7](#), [TABLE 10](#) define the function group for each permission. When the setting for the function group is changed it applies consistently to each permission. The initial value of the setting is the default listed in [TABLE 5](#).
- The device **must** have a set of function group settings for each untrusted MIDlet suite. The implementation **must not** offer a way to set function group settings for several MIDlet suites from the untrusted domain at the same time.
- A compliant implementation **must** be able to present the MIDlet suite name, version number and Function Group settings to the user for each installed MIDlet suite. An implementation **may** present any additional security related information to the user.

7.3.4 User Prompts and Notifications

The following rules **must** be followed in order to ensure informed user consent to MIDlet actions:

1. Any chargeable event generated by a MIDlet in the untrusted domain **must** be preceded by user notification in accordance with user permission settings. For example, showing the phone number (ISDN) or corresponding name the MIDlet is dialing, or the recipient (ISDN) of an SMS.
2. Any chargeable event in progress (for example, peer-to-peer connection the user is charged for) **must** be indicated to the user.
3. A MIDlet **must** obtain user approval to connect to the network, in accordance with user permission settings of the policy.
4. Any MIDlet permissions **should** be presented to the user in an intuitive, user-friendly manner.
5. A MIDlet **must not** be able to override security prompts and notifications to the user generated by the system or virtual machine.
6. A MIDlet **should not** be able to simulate security warnings to mislead the user.
7. For each MIDlet suite, the implementation **may** read requested permissions from the MIDlet-Permissions and MIDlet-Permissions-Opt attributes, notify the user which capability the application requires and prompt the user to accept or reject installation of the application.

7.4 MIDP 2.0 API Untrusted Policy

The MIDP 2.0 specification defines a number of restricted APIs for which permissions are defined and access may be restricted by the policy. The mappings of those permissions to the Function Groups is listed in [TABLE 6](#).

TABLE 6 MIDP 2.0 Function Groups Permissions

Function Group	Protocol	Permission
Net Access	http	javax.microedition.io.Connector.http
Net Access	https	javax.microedition.io.Connector.https
Net Access	datagram	javax.microedition.io.Connector.datagram
Net Access	datagram server (without host)	javax.microedition.io.Connector.datagramreceiver
Net Access	socket	javax.microedition.io.Connector.socket
Net Access	server socket	javax.microedition.io.Connector.serversocket

TABLE 6 MIDP 2.0 Function Groups Permissions

Function Group	Protocol	Permission
Net Access	ssl	javax.microedition.io.Connector.ssl
Local Connectivity	comm	javax.microedition.io.Connector.comm
Application Auto Invocation	All	javax.microedition.io.PushRegistry

Network Access Requirements

Untrusted applications must use the normal `HttpConnection` and `HttpsConnection` APIs to access web and secure web services. There are no restrictions on web server port numbers through these interfaces. The implementations augment the protocol so that web servers can identify untrusted applications. The following **must** be implemented:

- The implementation of `HttpConnection` and `HttpsConnection` **must** include a separate `User-Agent` header with the Product-Token "UNTRUSTED/1.0". `User-Agent` headers supplied by the application **must not** be deleted.
- The implementation of `SocketConnection` using TCP sockets **must** throw `java.lang.SecurityException` when an untrusted MIDlet suite attempts to connect on ports 80 and 8080 (http) and 443 (https).
- The implementation of `SecureConnection` using TCP sockets **must** throw `java.lang.SecurityException` when an untrusted MIDlet suites attempts to connect on port 443 (https).
- The implementation of the method `DatagramConnection.send` **must** throw `java.lang.SecurityException` when an untrusted MIDlet suite attempts to send datagrams to any of the ports 9200-9203 (WAP Gateway).
- The above requirements **should** be applied regardless of the API used to access the network. For example, the `javax.microedition.io.Connector.open` and `javax.microedition.media.Manager.createPlayer` methods **should** throw `java.lang.SecurityException` if access is attempted to these port numbers through a means other than the normal `HttpConnection` and `HttpsConnection` APIs.

7.5 Wireless Messaging Untrusted Policy

This section defines the untrusted policy for the Wireless Messaging API. The permissions in [TABLE 7](#) are defined in [WMA 1.1](#), Appendix D.

TABLE 7 Messaging Function Group Permissions

Function Group	Permission
Messaging	javax.microedition.io.Connector.sms
Messaging	javax.microedition.io.Connector.cbs
Messaging	javax.wireless.messaging.sms.send
Messaging	javax.wireless.messaging.sms.receive
Messaging	javax.wireless.messaging.cbs.receive

User Prompting

The untrusted messaging policy requires that the user **must** be provided with the destination phone number or a corresponding name when prompting for the messaging function group. If a single API call is mapped to multiple messages (i.e. the implementation supports disassembly or reassembly), then the implementation **must** provide the user with the number of messages that are to be sent. This requirement is to ensure that the user always understands the network costs.

Interaction Modes

If the interaction mode is *oneshot* for the messaging function group the connections are assigned the *blanket* mode as shown in [TABLE 8](#).

TABLE 8 Interaction Modes for Connections

Permission	Function Group Interaction Mode	Permission Interaction Mode
javax.microedition.io.Connector.sms	Oneshot	Blanket
javax.microedition.io.Connector.cbs	Oneshot	Blanket

If the interaction mode is *oneshot* for the messaging function group, then the mode for the permissions for receiving messages is *blanket* as shown in [TABLE 9](#). Also, the permissions for sending messages are the same as shown in [TABLE 9](#). Thus the interaction mode is *oneshot* for each message sent by the MIDlet suite. When the

interaction mode is *no*, it applies to all the individual permissions within the messaging function group. These requirements apply regardless of the intended recipient, whether it is a user or another application.

TABLE 9 Interaction Modes for Sending and Receiving Messages

Permission	Function Group Interaction Mode	Permission Interaction Mode
<code>javax.wireless.messaging.sms.send</code>	Oneshot	Oneshot
<code>javax.wireless.messaging.sms.receive</code>	Oneshot	Blanket
<code>javax.wireless.messaging.cbs.receive</code>	Oneshot	Blanket

7.6 Mobile Media API Untrusted Policy

TABLE 10 Mobile Media API Function Group Permissions

Function Group	Permission
Multimedia Recording	<code>javax.microedition.media.control.RecordControl</code>
Multimedia Recording	<code>javax.microedition.media.control.VideoControl.getSnapshot</code>

Implementations **must** ensure that I/O access from the Mobile Media API follows the same security requirements as the Generic Connection Framework, as specified in the package documentation for `javax.microedition.io`. Example methods include `javax.microedition.media.Player.start`, `javax.microedition.media.Player.prefetch`, etc. When these methods are used to fetch the content for the player via an HTTP connection, the implementation **must** enforce the security requirements specified for HTTP.

