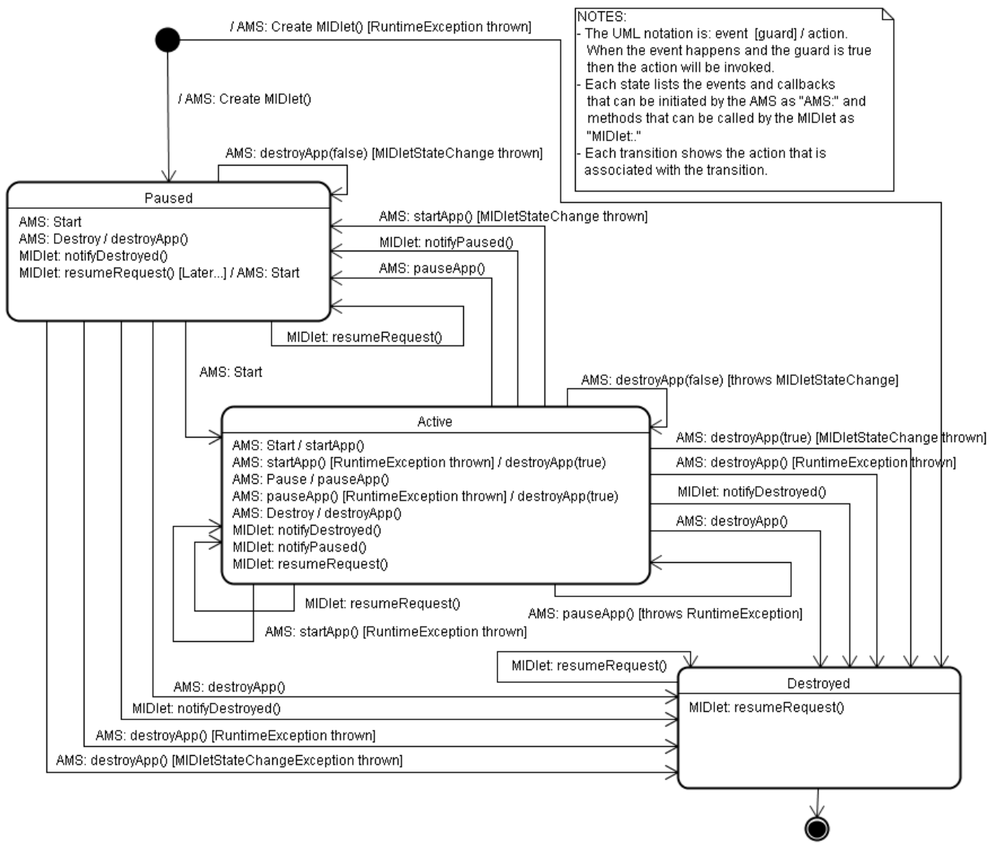
JSR-118 Maintenance Release 2.1 Change Log

Updated 7-Apr-2006

Number	Change	Reference	Status
1	The minimum size (storage capacity) of a TextField or a TextBox MUST NOT be less than 1000 characters.	javax.microedition.lcdui.TextField javax.microedition.lcdui.TextBox	REJECTED for 2.1
2	Make LCDUI layout directives (MIDP 2.0 specification pages 292-294) mandatory	MIDP 2.0 specification pages 292-294	ACCEPTED for 2.1
3	The behavior of the LayerManager.insert() method MUST comply with the following description:insert(Layer, int) **Declaration:** public void insert(javax.microedition.lcdui.game.Layer l, int index) **Description:** Inserts a new Layer in this LayerManager at the specified index. Inserting a Layer that was previously added to this LayerManager is equivalent to first removing it with LayerManager.remove() and then adding it with the insert() method at the specified index. Conditions for throwing IndexOutOfBoundsException are checked before LayerManager.remove() is called. **Parameters:** l – The Layer to be inserted index – The index at which the new Layer is to be inserted **Throws:** NullPointerException – if the Layer is null IndexOutOfBoundsException: • If the index is less than 0 • If the index is greater than the number of Layers already added to this LayerManager and the Layer has not been added to this LayerManager yet • If the index is greater than the number of Layers already added to this LayerManager minus one and the Layer has already been added to this LayerManager.	javax.microedition.lcdui.LayerManager	ACCEPTED for 2.1
4	A StringItem that has item commands and the appearance mode Item.PLAIN MUST always be presented as a StringItem with added command(s) and appearance mode Item.HYPERLINK.	javax.microedition.lcdui.Item javax.microedition.lcdui.StringItem	ACCEPTED for 2.1
5	If the device keypad can be used in different modes, and some modes restrict user access to certain keys, or if the device has several different keypads that are not meant to used simultaneously, a mobile handset implementation is NOT REQUIRED to ensure that the mapping between key codes and game action does not change during the execution of the application. Consequently, applications SHOULD NOT use the Canvas.getKeyCode() method to learn which key is assigned to a certain game action. Instead, applications SHOULD use the Canvas.getGameAction() method to check which game action is associated with a given key code.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
6	Many MIDP LCDUI graphical components can contain text (that is, an alphanumeric string) that is shown to the user. Examples of such components are List, TextBox, Alert, StringItem, Form, and Item. An implementation often needs to truncate such visible text because it does not fit in the designated space of a given UI component. In this case, an implementation MUST use an appropriate visual indication (for example an ellipsis symbol) to signal the user that the text is truncated. The actual symbol or symbols used to represent the truncated text depends on the locale that is currently selected in the device. However, the visual indication SHOULD be consistent with the visual indication used in the device's native UI.	javax.microedition.lcdui.List javax.microedition.lcdui.Alert javax.microedition.lcdui.Item javax.microedition.lcdui.Form javax.microedition.lcdui.StringItem	ACCEPTED for 2.1
7	Canvas pointer events MUST be supported, if the underlying hardware supports this feature. In this case, the method Canvas.hasPointerEvents()SHALL always return true.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
8	Canvas pointer motion events MUST be supported, if the underlying hardware supports this feature. In this case the method Canvas.hasPointerMotionEvents() SHALL always return true.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
9	Canvas repeat events MUST be supported. The method Canvas.hasRepeatEvents() SHALL always return true.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
10	Double-buffered graphics MUST be supported. The method Canvas.isDoubleBuffered() SHALL always return true.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
11	A device can have a 3-way or 5-way jog dial as a control mechanism. A 3-way jog dial is usually a wheel that rotates in two directions (to indicate scrolling) and	javax.microedition.lcdui.Canvas	ACCEPTED

	can also be pressed (to indicate a selection). A 5-way jog dial is typically similar to a 3-way jog dial with the added possibility to tilt the wheel sideways. A jog dial wheel might have the ability to be rolled several steps in each direction. Alternatively, a jog dial wheel might only have the ability to be rotated by a limited angle, returning to the base position when released. When MIDP is implemented on a device with a jog dial, the requirements are as follows: • An implementation MUST generate key press events in response to jog dial movements. An implementation MAY reuse key codes assigned to device keys with functions similar to jog dial functions (for example, key codes assigned to cursor keys can also be used for jog dial events). However, an implementation MAY choose to use unique key codes for jog dial events. • An implementation MUST map key press events generated by jog dial movements to game actions. An application can learn game actions via the Canvas.getGameAction() method. The mapping of game actions MUST always be natural in relation to the spatial orientation of the device (that is, mapping MUST be natural from the user point of view. For example, if the jog dial is rolled upwards, UP events are generated). This also means that an implementation MAY need to deliver different key events and game actions depending on the spatial orientation of the device. For example, an application running in a mode that requires a certain orientation of the device can get different key codes than an application running in another orientation. In the same vein, a clamshell device MAY generate different jog dial events when the clamshell is in OPEN or CLOSED mode. • A 3-way jog dial MUST generate key events with key codes mapping to game actions UP, DOWN and FIRE. • A 5-way jog dial MUST generate key events with key codes mapping to game actions UP, DOWN, LEFT, RIGHT and FIRE. • A jog dial MAY also be used to control high level UI objects. **Note**: Depending on the mechanics of the jog wheel, the implementation is not necessarily able to generate key repeat events for some movements (for example, when the wheel is rotated).		for 2.1
12	If Canvas is in full-screen mode, the following requirements apply: • If CommandListener is set for a Canvas, an implementation MUST display the commands in the User Interface when a key normally reserved for the delivery of commands is pressed and deliver the selected command. • If no CommandListener is set for a Canvas, an implementation MUST deliver key code events for key presses that are otherwise reserved for the delivery of commands.	javax.microedition.lcdui.Canvas	ACCEPTED for 2.1
13	Availability of various text input modes (for example, predictive input and numbers-only input) SHOULD be consistent across Java and native applications. This means, for example, that if predictive text input mode is available in native applications, it SHOULD also be available in Java applications.	javax.microedition.lcdui.TextField javax.microedition.lcdui.TextBox	ACCEPTED for 2.1
14	The creation of Image objects (regardless of the format) MUST at least support: sizes equal to (screen width) by (screen height) by (colour depth in bits) or 262144 bits (128x128x16 bits = 32 kB), whichever value is greater. Note that the internal representation of an Image object SHOULD hold at least 16 bits of colour/transparency data per pixel.	javax.microedition.lcdui.Image	REJECTED for 2.1
15	Correct the first example on page 57 of the MIDP 2.0 specification that contains an error (com0 in the first line MUST be COM1). Here is the corrected version of the code: `CommConnection cc = (CommConnection)` `Connector.open(“comm:COM1;baudrate=19200”);` `int baudrate = cc.getBaudRate();` `InputStream is = cc.openInputStream();` `OutputStream os = cc.openOutputStream();` `int ch = 0;` `while(ch != 'Z') {` `os.write(ch);` `ch = is.read();` `ch++;` `}` `is.close();` `os.close();` `cc.close();` Also, “Recommended Port Naming Convention” on pages 57-58 is replaced with the following requirements: Port Naming Requirements: Ports have to be named consistently among the implementations of this class. Therefore, VM implementations MUST follow the following convention: Port names MUST contain a text abbreviation indicating port capabilities followed by a sequential number for the port. Port numbering MUST start from 1 without any leading zeros, and numbering MUST be sequential (without any gaps). The following device name types MUST be used:	javax.microedition.io.CommConnection Pages 57-58.	ACCEPTED for 2.1

	• COM#, where COM is for RS-232 ports and # is a number assigned to the port • IR#, where IR is for IrDA IRCOMM ports and # is a number assigned to the port • USB#, where USB is for USB ports and # is a number assigned to the port • BT#, where BT is for Bluetooth Serial Port Profile ports and # is a number assigned to the port. This naming scheme allows API users to generally determine the type of port to use. For example, if an application needs to beam a piece of data, the application could look for IR# ports for opening the connection. The alternative is a trial-and-error approach with all available ports. If the implementation cannot determine a port type, the same naming convention as for an RS-232 port MUST be used.		
16	Every serial port name included in the String value returned by the method System.getProperty("microedition.commports") MUST be accessible via the javax.microedition.io.CommConnection interface.	javax.microedition.io.CommConnection	ACCEPTED for 2.1
17	In every protocol, the AllowedSender field MUST match with the appropriate address field of the incoming event. The address field to use and the exact syntax and semantics of the address depend on the protocol. However, the address and the filter MUST be compared by exact string matching where the strings are compared character by character and the characters need to match exactly except as allowed by the two wildcard characters: asterisk (*) and question mark (?). The semantics of these wildcard characters are defined in the MIDP 2.0 specification.	javax.microedition.io.PushRegistry javax.microedition.io.SecureConnection javax.microedition.io.SocketConnection javax.microedition.io.HttpsConnection javax.microedition.io.ServerSocketConnection	ACCEPTED for 2.1
18	The following network communication protocols MUST be supported, that is, provide an implementation of the following Java ME interfaces: • javax.microedition.io.SocketConnection • javax.microedition.io.SecureConnection • javax.microedition.io.HttpsConnection	javax.microedition.io.SecureConnection javax.microedition.io.SocketConnection javax.microedition.io.HttpsConnection	REJECTED for 2.1
19	For javax.microedition.io.HttpsConnection and javax.microedition.io.SecureConnection interfaces the SSL V3 protocol MUST be supported as the secure protocol. Other secure protocols, such as TLS and WTLS, MAY also be supported.	javax.microedition.io.SecureConnection javax.microedition.io.HttpsConnection	ACCEPTED for 2.1
20	The following diagram illustrates the requirements related to the MIDlet lifecycle, as defined in the MIDP Specification:	Page 439	ACCEPTED for 2.1



21	The MIDP specification requires that the application descriptor contain the following attributes: • MIDlet-Name • MIDlet-Version • MIDlet-Vendor • MIDlet-Jar-URL • MIDlet-Jar-Size In addition, to improve the user experience (by rejecting incompatible suites without needing to download the JAR file), the application descriptor SHOULD contain the following attributes: • MicroEdition-Profile • MicroEdition-Configuration • MIDlet-Permissions	javax.microedition.io.PushRegistry Page 433-435	ACCEPTED for 2.1
22	The following JAD / JAR manifest attribute is defined to support specifying the expected runtime execution environment: **Attribute name** | **Attribute description**	Page 431-435	ACCEPTED for 2.1

	<table><tr><td>Runtime-Execution-Environment</td><td>This is an optional attribute that indicates the runtime execution environment REQUIRED by the application. This attribute MAY have the value MIDP.CLDC. If the MIDlet suite does not specify the attribute, the implicit default value is MIDP.CLDC. When the value of the Runtime-Execution-Environment attribute is MIDP.CLDC, the behavior of the implementation MUST adhere to the requirements presented in the following more detailed description. Additional values for this attribute MAY be defined in the future.</td></tr></table> A mobile handset implementation MUST support the Runtime-Execution-Environment attribute. **Runtime-Execution-Environment** When an application defines the Runtime-Execution-Environment attribute value MIDP.CLDC or does not define this attribute, the behavior of the mobile handset implementation during execution of this application MUST adhere to the following requirements: • The set of supported APIs and the API behavior, as well as the behavior of the underlying virtual machine, MUST comply with the CLDC 1.1 specification. • A mobile handset implementation MAY alternatively use the CDC configuration of Java ME. However, if CDC is used as the underlying configuration, applications running on top of this platform MUST see an environment that is semantically and functionally equivalent to a CLDC 1.1 platform. CDC-specific APIs or CDC-specific behavior MUST NOT be exposed to applications or application developers. • When an application defines the Runtime-Execution-Environment attribute value MIDP.CLDC the application MUST also define a CLDC platform in the MicroEdition-Configuration attribute. • If an application defines a value for the Runtime-Execution-Environment or MicroEdition-Configuration attribute that is not supported by the implementation, the application MUST NOT be installed. All mobile handset implementations MUST support the "MIDP.CLDC" value for the Runtime-Execution-Environment attribute.		
23	When an uncaught exception occurs in a thread, the thread MUST be stopped in accordance with the Java Language Specification. In addition, further handling of the uncaught exception MAY occur as follows: • If an implementation can identify the MIDlet that created the thread (the MIDlet to which the thread belongs), the MIDlet.destroyApp(false) method of this MIDlet MAY be called. • If an implementation can only identify the MIDlet suite that created the thread (the MIDlet suite to which the thread belongs), the MIDlet.destroyApp(false) methods of all MIDlets from this suite MAY be called. • If an implementation cannot identify either a MIDlet or a MIDlet suite that owns the offending thread, MIDlet.destroyApp(false) methods of all MIDlets executing in this VM instance MAY be called. In all of these cases, the behavior after a call to MIDlet.destroyApp(false) is as follows: • If a MIDlet attempts to recover by throwing javax.microedition.midlet.MIDletStateChangeException, the implementation MAY allow the MIDlet to continue running. • If a call to MIDlet.destroyApp(false) results in a different exception of any kind, the implementation MUST terminate the MIDlet.	javax.microedition.midlet.MIDlet javax.microedition.midlet.MIDletStateChangeException	ACCEPTED for 2.1
24	In case of user-initiated OTA provisioning, after a MIDlet suite is successfully installed on the device and if the suite contains any MIDlets (that is, one or more MIDlet-<n> attributes are in the JAD file or JAR manifest), an implementation MUST prompt the user to specify whether to launch a MIDlet from the suite. If the user approves launching a MIDlet and the MIDlet suite contains only one MIDlet, that specific MIDlet MUST be launched. If the suite contains several MIDlets, an implementation MUST give the user the opportunity to choose which MIDlet to launch. After the selection, the chosen MIDlet MUST be launched. These requirements apply only to MIDlets that are listed in the JAD file or in the JAR manifest by means of MIDlet-<n> attributes.	Page 12-13	ACCEPTED for 2.1
25	If an implementation displays MIDlet icons (using associated MIDlet-Icon attribute), the following requirements apply: • The application manager MUST always be able to display a MIDlet-Icon of 16 by 16 pixels in the application list. • The application manager SHOULD always be able to display a MIDlet-Icon of 24 by 24 pixels in the application list. • A MIDlet-Icon larger than 24 by 24 MAY be displayed properly in full size in the application list. • A MIDlet-Icon MUST NOT be scaled down in the application list, but the application manager MAY make use centered cropping to fit the icon into the list.	Page 433	ACCEPTED for 2.1
26	If a MIDlet suite contains multiple MIDlets, an implementation MUST make the user aware of the MIDlets and associated RMS record stores that will be removed, and the user MUST be given the chance to abort this action.	Page 463	ACCEPTED for 2.1

27	In MIDlet suite installations and deletions, if the status report cannot be sent, or if the server reply is not received, an implementation MUST attempt to resend the report at least 5 times. Attempts to resend the status report MUST NOT be performed after a response from the server is received. The precise conditions for resending the status report (both for installations and deletions) are defined in the MIDP specification.	Page 16	ACCEPTED for 2.1
28	An implementation MUST allow a MIDlet to create a minimum of 10 simultaneously running threads.	javax.microedition.midlet.MIDlet	REJECTED for 2.1
29	An implementation MUST provide the following minimum storage capacity for JAD and JAR files to support OTA download and installation of MIDlet suites: • At least 10 kB for a JAD file • At least 300 kB for a JAR file This requirement does not apply in cases where, because the device is occupied with downloading other content (including other MIDlet suites), there is insufficient memory to store another MIDlet suite.	Page 13	REJECTED for 2.1
30	An implementation MUST support at least 512 attributes in both JAD files and manifests. A mobile handset implementation MUST support JAD attribute value size of at least 2048 bytes in both JAD files and manifests. A mobile handset implementation MUST support attribute name size of 70 bytes in both JAD files and manifests. 70 bytes is the maximum length for an attribute name defined in the JAR File Specification [JAR].	Page 13	REJECTED for 2.1
31	An implementation MUST support MIDlet suites containing any number of MIDlets between one and five (inclusive). A mobile handset implementation MAY support MIDlet suites containing more than five MIDlets. This requirement applies to suite installation and MIDlet presentation in the device UI. It does not apply to the ability to run MIDlets simultaneously.	Page 12	REJECTED for 2.1
32	An implementation MUST be able to honor requests by MIDlet suites for an RMS data size of at least 64 kB. This means that if a MIDlet suite requests to reserve 64 kB (or less) for its RMS data (using its MIDlet-Data-Size attribute), an implementation MUST reserve the requested amount of memory for RecordStores created by MIDlets from this suite. This requirement does not apply in cases where insufficient memory is available to honor the request because the device memory is occupied with other data (including RecordStores of other MIDlet suites). However, if an implementation honors a request for a certain size of RMS data, this amount of memory MUST be available to MIDlets from the suite during the entire time the suite is installed on the device. More precisely, a mobile handset implementation MUST ensure the following: • A MIDlet suite requesting the RMS data size less than or equal to 64 kB in the MIDlet-Data-Size attribute can be successfully installed and MIDlets from that suite can run. • If a MIDlet from the suite creates a single (empty) RecordStore, the RecordStore.getSizeAvailable() method returns a value greater than or equal to the value of the MIDlet-Data-Size attribute. • At any time, a MIDlet from the suite can store a single byte array of at least RecordStore.getSizeAvailable() bytes using the RecordStore.addRecord() method. In particular, a single byte array size of the value of the MIDlet-Data-Size attribute can be stored in an empty record store.	Page 15 Page 463	REJECTED for 2.1
33	An implementation MUST permit a MIDlet suite to create at least 10 independent RecordStores.	Page 15 Page 463	REJECTED for 2.1
34	The method MIDlet.platformRequest() MUST support at least the following types of requests: • URL pointing to a web site (http://... , according to RFC 2716). In this case, the implementation MUST either launch the native browser application, bring it to the foreground while keeping the MIDlet running in the background, or wait until the MIDlet exits and then start the browser application. In the former case the MIDlet.PlatformRequest() method MUST return false; in the latter case it MUST return true. • URL pointing to a phone number (tel://<number>, according to RFC 3966). In this case, the implementation MUST either launch the native call application and bring it to the foreground while keeping the MIDlet running in the background, or wait until the MIDlet exits and then start the call application. In the former case the MIDlet.PlatformRequest() method MUST return false; in the latter case it MUST return true. If a URL passed in to the platformRequest() method points to a MIDlet suite residing in the network (either to a JAD or a JAR file), an implementation MUST interpret the method call as a request to install the suite. In this case, the normal MIDlet suite OTA provisioning process MUST be performed, and the user MUST be allowed to control the process (including cancelling the download, the installation, or both). If the MIDlet suite being installed is an update of an installed MIDlet suite, and if some MIDlets from that suite are currently being executed, the platform MUST stop all such MIDlets before performing the update.	Page 447	REJECTED for 2.1
35	The MIDP subset of MMAPI (classes from the javax.microedition.media package) MUST comply with the MMAPI specification version 1.1 (or later).	javax.microedition.media.Player.getMediaTime() javax.microedition.media.control.ToneControl	ACCEPTED for 2.1

36	The microedition.profiles system property MUST NOT contain different versions of the same profile. In particular, MIDP 1.0 and MIDP 2.0 MUST NOT both be listed.	Page 35	ACCEPTED for 2.1
37	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.	Chapter 10	REJECTED for 2.1
38	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(javaio.InputStream stream)` `public static Image createImage(String name)` 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. 1 *ISO/IEC JPEG* *ITU-T Recommendation T.81: "Information technology; Digital compression and coding of continuous-tone still images: Requirements and guidelines" 09/92.* `http://www.w3.org/Graphics/JPEG/itu-t81`	Chapter 1 and 8	ACCEPTED for 2.1
39	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.	Chapter 6	REJECTED for 2.1
40	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."	Chapter 6	REJECTED for 2.1
41	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(javaio.InputStream stream)` `public static Image createImage(String name)`	Chapter 8	REJECTED for 2.1

	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.		
42	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: 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 / 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	Chapter 8	ACCEPTED for 2.1
43	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`.	Chapter 8	ACCEPTED for 2.1
44	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	Chapter 8	REJECTED for 2.1

	no other security mechanism (such as described in Chapter 7) is present, the `PushRegistry` Alarm function **must not** be allowed without explicit user permission.		
45	Recommended Security Policy – corrected typos and minor edits found since 2.0.1	Recommended Security Policy Addendum	ACCEPTED for 2.1