IBM Software Group | Lotus Expeditor 6.1.1 Education

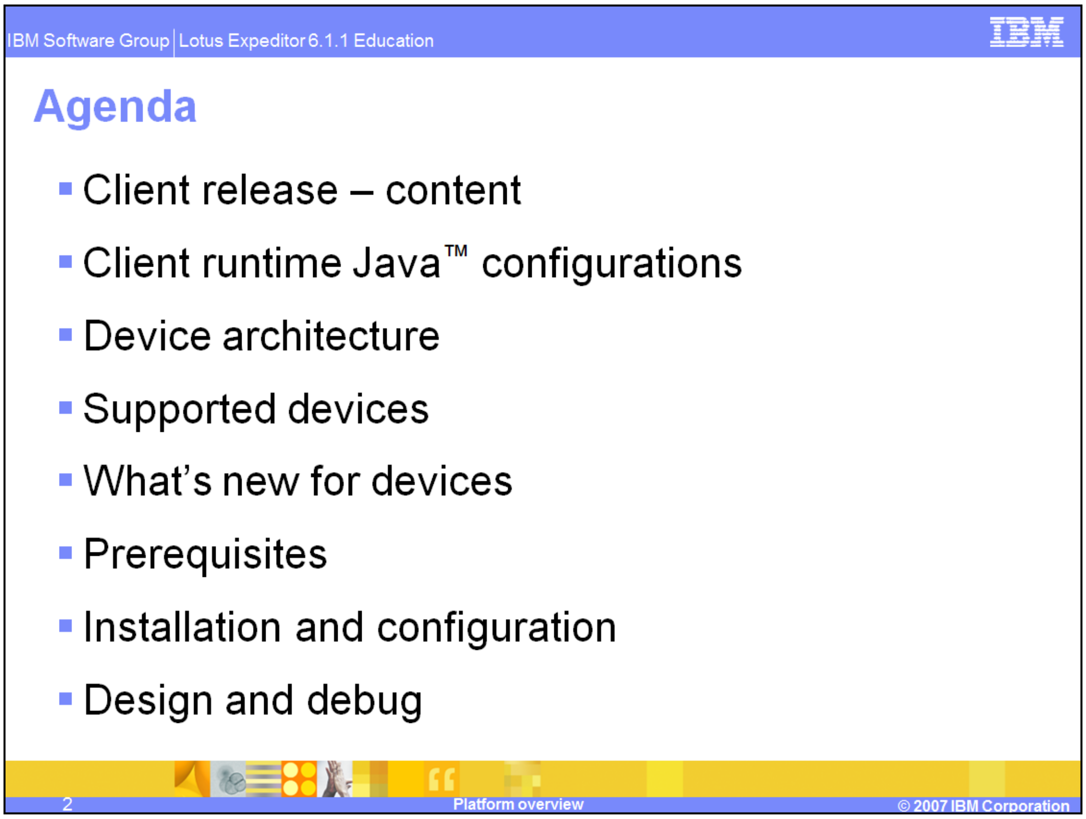
IBM Lotus® Expeditor Client for Devices

Platform overview

Lotus software

e business on demand software

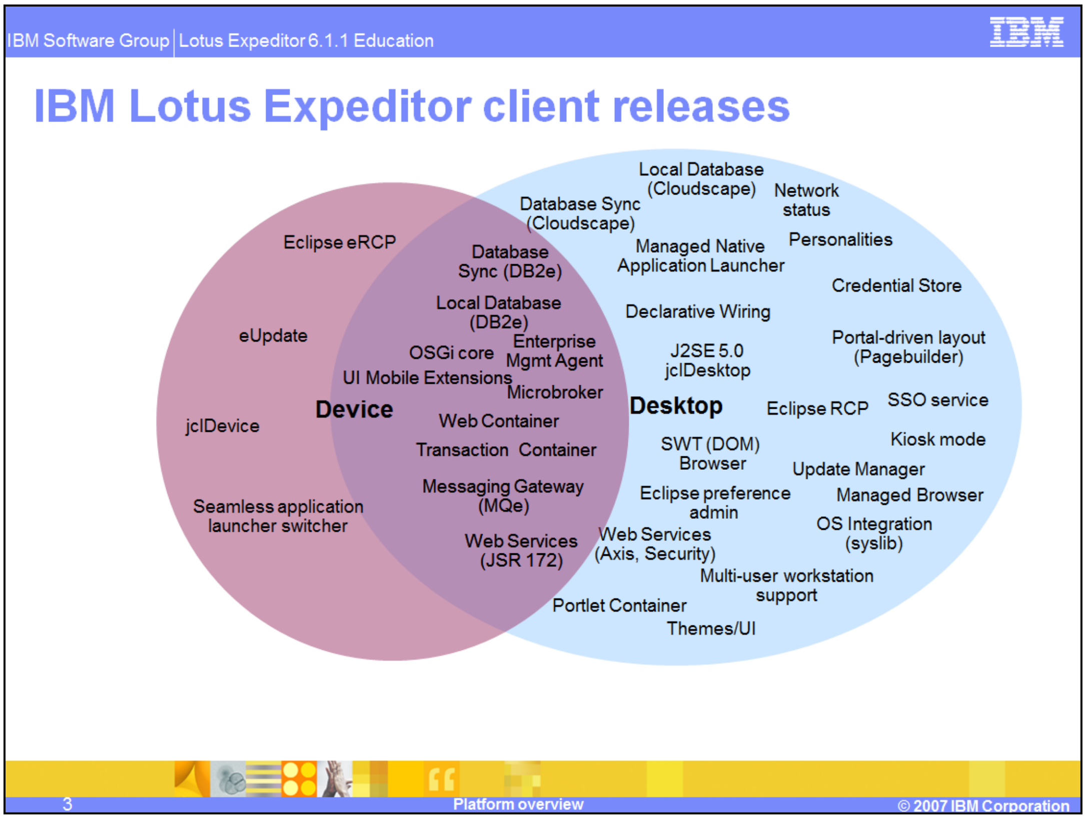
This presentation introduces Lotus Expeditor Client for Devices Version 6.1.



Agenda

- Client release – content
- Client runtime Java™ configurations
- Device architecture
- Supported devices
- What's new for devices
- Prerequisites
- Installation and configuration
- Design and debug

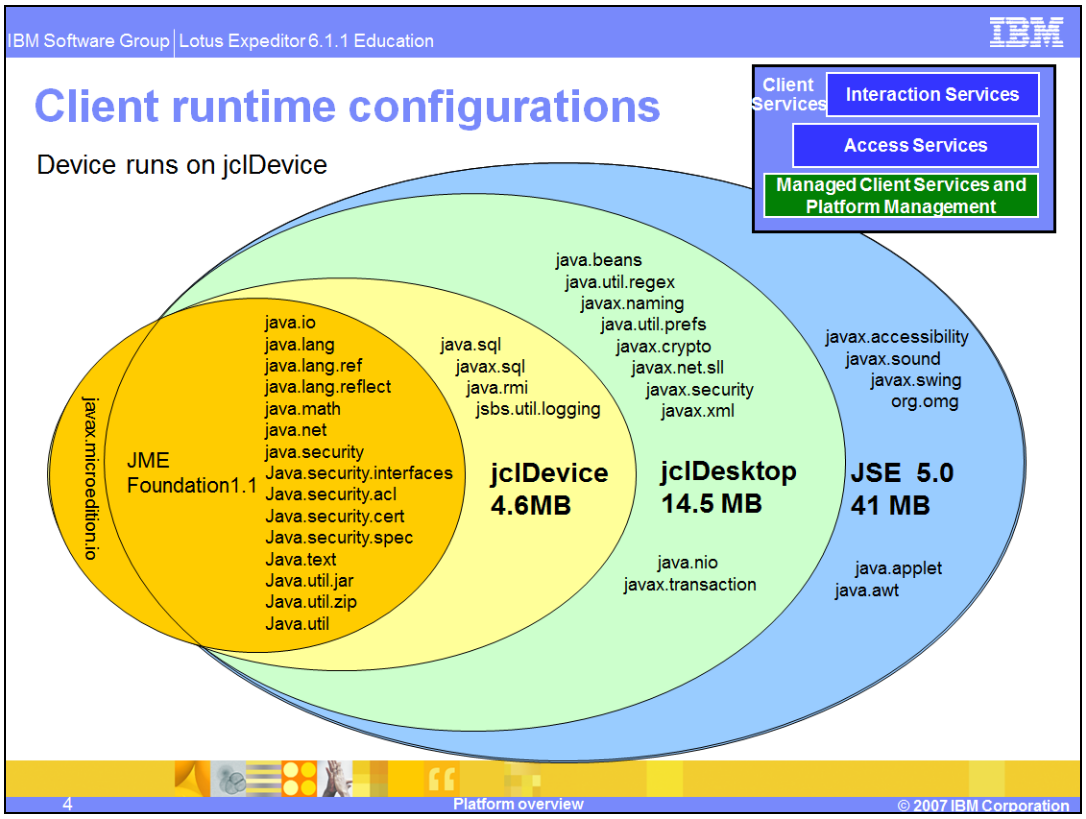
The agenda of this presentation is to provide an overview of Lotus Expeditor Client for Devices release, along with describing the Client Runtime Java Configurations, the Device Architecture, the currently supported devices, the newly added features in the device release, prerequisites, how to install and configure your device, and finally design and debug considerations.


IBM Software Group | Lotus Expeditor 6.1.1 Education

IBM Lotus Expeditor client releases

3 Platform overview © 2007 IBM Corporation

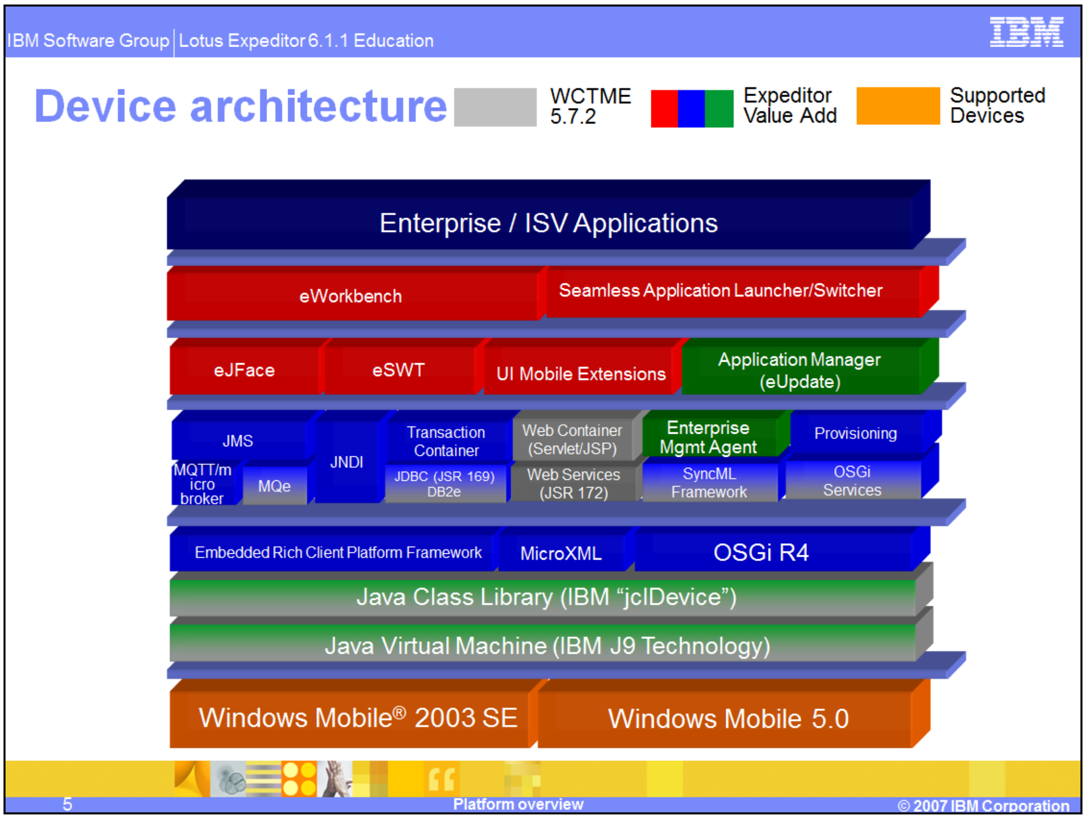
The Lotus Expeditor Client Release includes two configurations. One is desktop; the other is device. Both releases have some components in common, like Database Sync, Enterprise Management Agent, Lotus Expeditor micro broker, Web Container, Transaction Container, MQe, and Web Services. These are mainly targeted for end-to-end Client Services solutions. More information about the desktop configuration is available in a separate presentation.



Client runtime configurations

Device runs on jclDevice

jclDevice is a customized Java Runtime Environment that provides several extra packages in addition to CDC/Foundation 1.1. These packages were added to support some advanced Client Services on devices, like Web Container, logging, and JNDI. Although these extra packages are available to applications, their use is not recommended because they are included only to support Expeditor components and may not be included in future releases.
The desktop configuration uses jclDesktop, which includes more packages, but is still a subset of JSE.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Device architecture

▨ WCTME 5.7.2 🟥🟦🟩 Expeditor Value Add 🟧 Supported Devices

5 Platform overview © 2007 IBM Corporation

Here we put all of these capabilities and technologies together in one diagram, which represents the DNA of our client environments.

•The underlying base for all other components are the Java Virtual Machine, Java Class Library, and OSGi Framework.

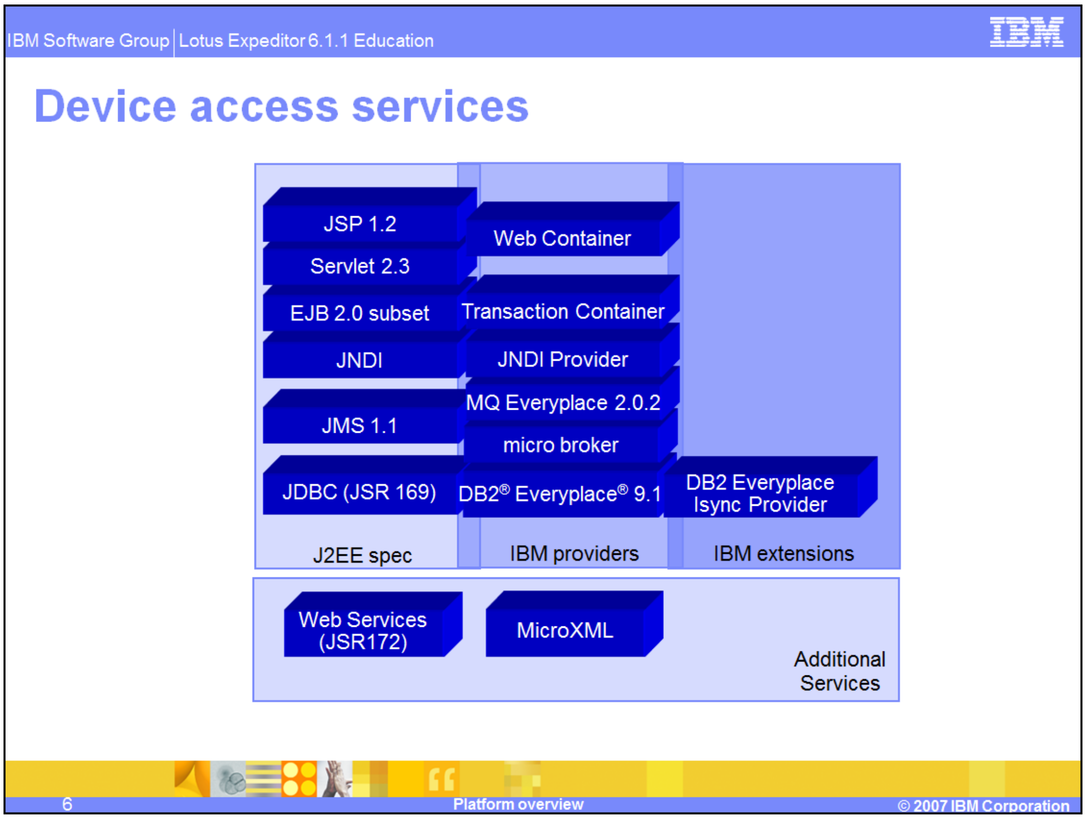
•The Platform Management components are the Enterprise Management Agent and the Application Manager.

•The Access Services include:

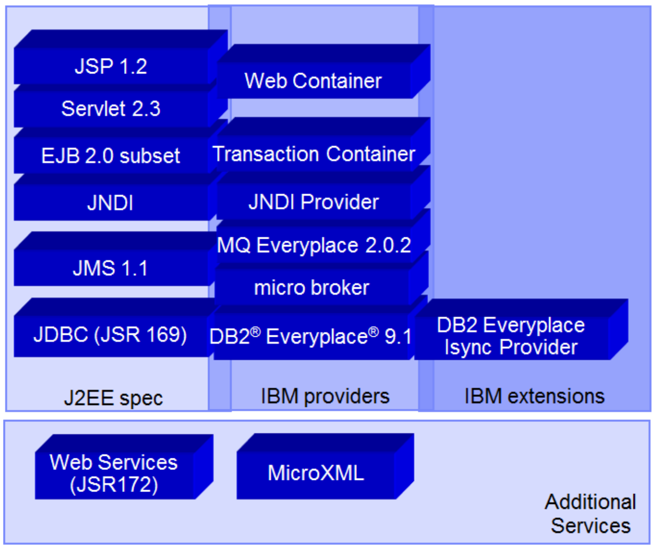
- The Lotus Expeditor micro broker, MQ Telemetry Transport, MQe, and JMS for assured transactional messaging
- JDBC and DB2e for relational database synchronization and local store
- Transaction Container, Web container, and XML parsing for application execution. This includes EJB, Servlet, JSP, and Web services.
- SyncML Framework and other OSGi services for standardized application interactions

•The Interaction Services layer includes eSWT, eJFace, and Mobile Extensions

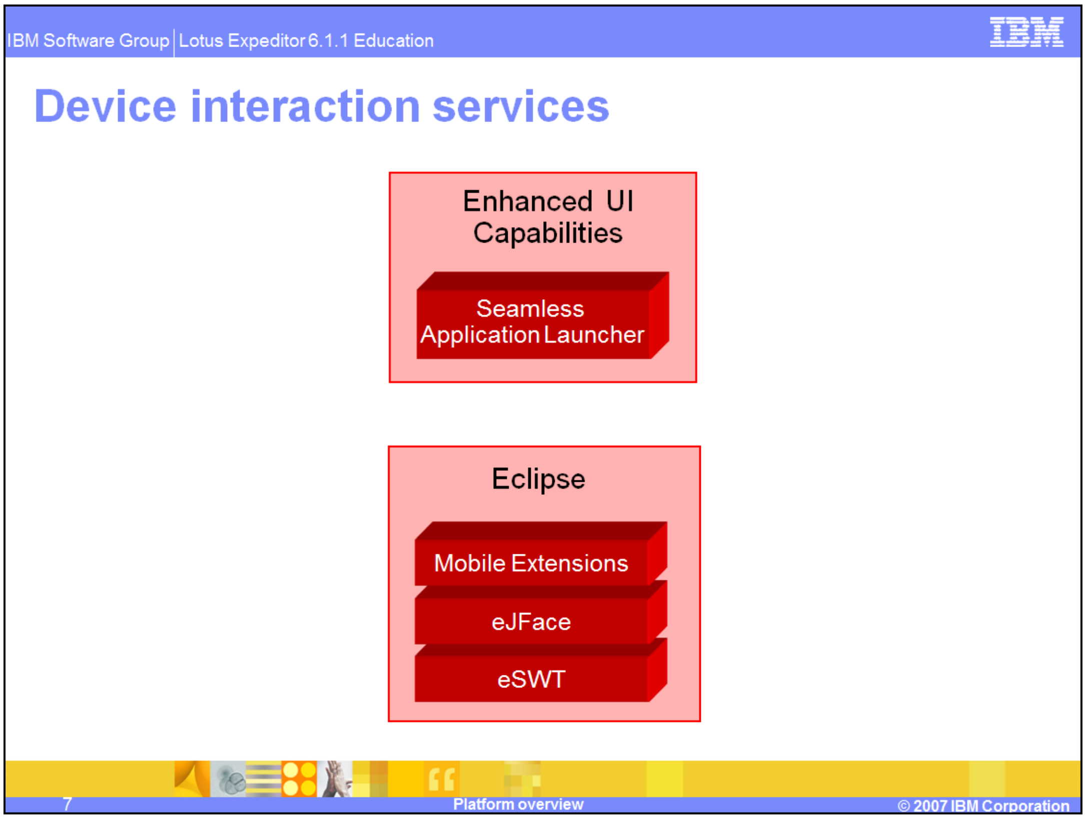
The Expeditor client provides a cohesive and consistent application platform for intermittently connected systems and devices.



Device access services



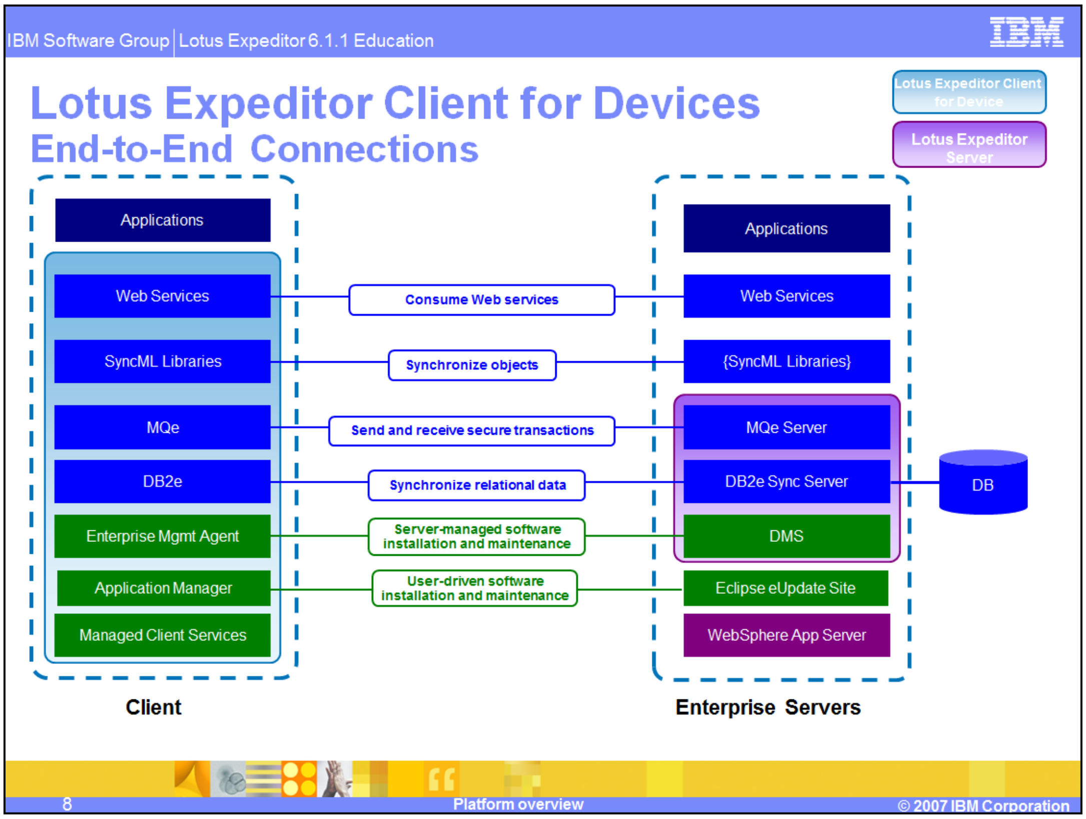
The device platform has a subset of the services in the desktop platform. Some services are from the J2EE specification, like JSP/Servlet, EJB, JNDI, JMS and JDBC driver. Others are IBM-provided components like Web Container, Transaction Container, JNDI provider, MQe, micro broker and DB2 Everyplace.



Device interaction services

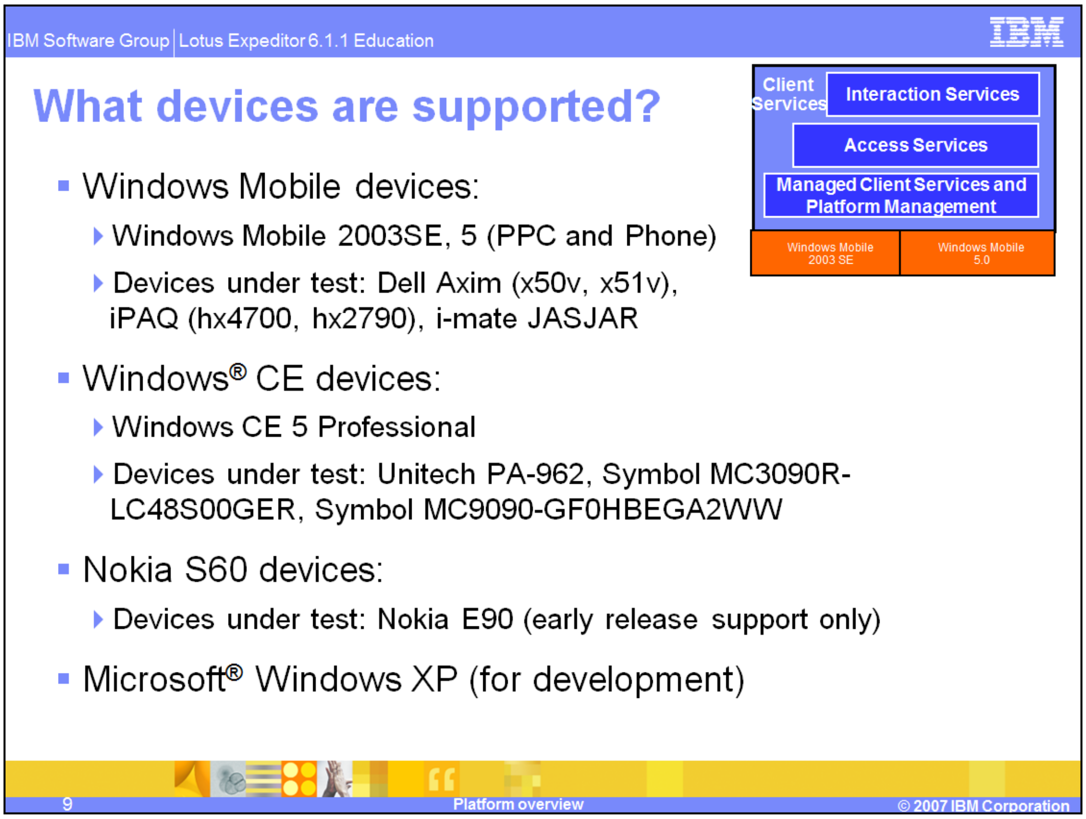
User interface APIs are subsets of the desktop SWT and JFace components. In addition, Mobile Extensions provides unique support for mobile devices. This helps abstract the differences among varied device types. This component is also provided in the desktop configuration so that embedded applications will function on the desktop runtime as well.

The seamless application launcher simplifies using Java applications on devices by making them start and act just like native applications.

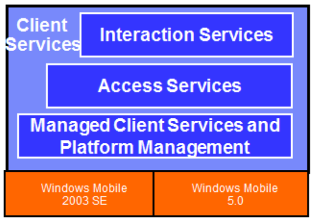


Lotus Expeditor Client for Devices
End-to-End Connections

This diagram shows how components in the client runtime interact with components on Enterprise servers to provide “Line of Business” data to clients that are not always connected.

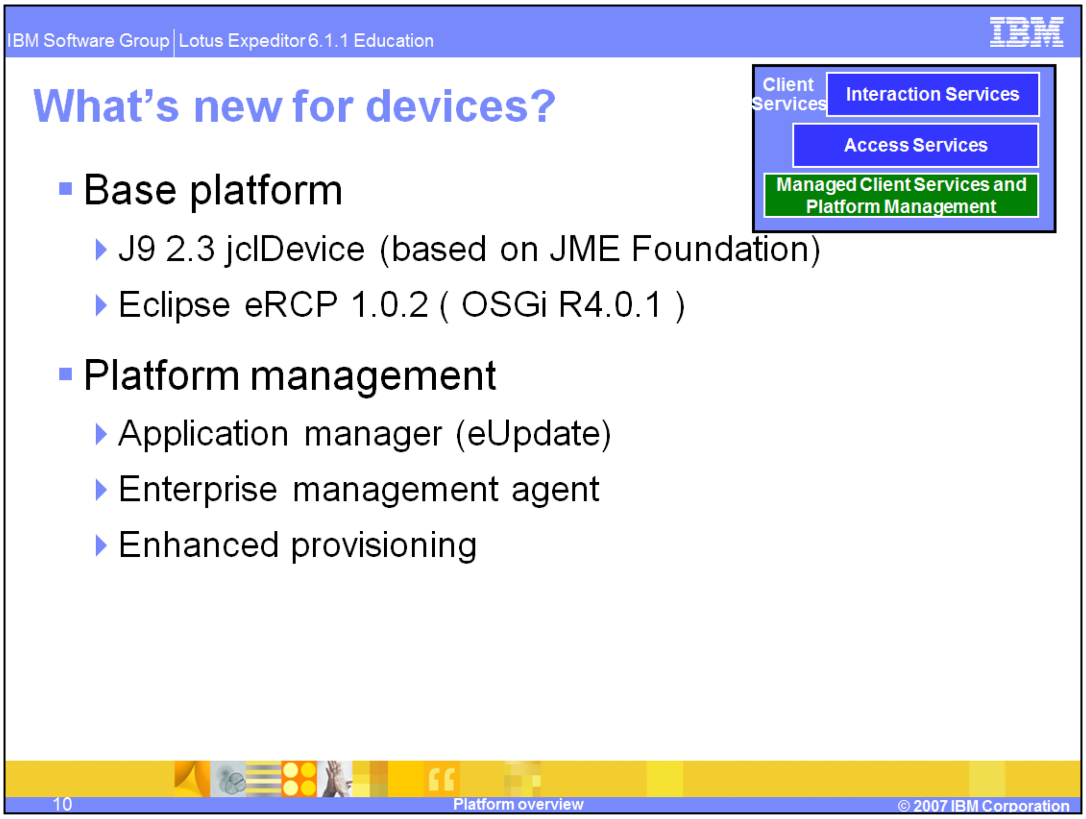


What devices are supported?



- Windows Mobile devices:
  - Windows Mobile 2003SE, 5 (PPC and Phone)
  - Devices under test: Dell Axim (x50v, x51v), iPAQ (hx4700, hx2790), i-mate JASJAR
- Windows® CE devices:
  - Windows CE 5 Professional
  - Devices under test: Unitech PA-962, Symbol MC3090R-LC48S00GER, Symbol MC9090-GF0HBEGA2WW
- Nokia S60 devices:
  - Devices under test: Nokia E90 (early release support only)
- Microsoft® Windows XP (for development)

Microsoft Windows Mobile 2003 Second Edition and Windows Mobile 5 are supported. We recommend that a device have at least 16 MB of file system space and 16 MB of RAM available. The device runtime is fully tested on Dell Axim (x50v and x51v), HP iPAQ 4700 and 2790, and iMate JASJAR. We also provide a Resource Checker to reveal device capabilities prior to installing Expeditor. Microsoft Windows XP is supported for development.

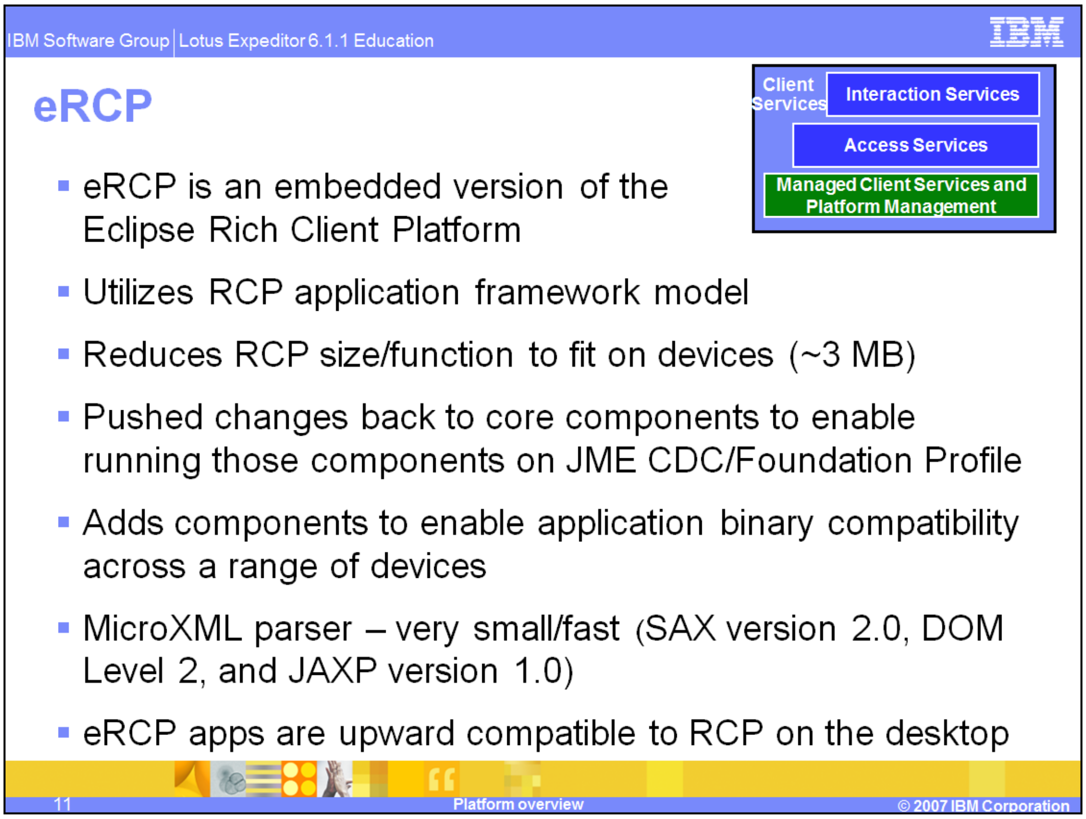


What's new for devices?

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- Base platform
  - J9 2.3 jclDevice (based on JME Foundation)
  - Eclipse eRCP 1.0.2 (OSGi R4.0.1)
- Platform management
  - Application manager (eUpdate)
  - Enterprise management agent
  - Enhanced provisioning

We include an IBM customized Java Runtime Environment – jclDevice, which is based on JME Foundation 1.1. Eclipse eRCP 1.0 is used as the base runtime platform. Application Manager is based on the Eclipse Update Manager. Enterprise Management Agent allows the device to be remotely managed by an administrator. Finally, enhanced provisioning allows the Enterprise Management Agent to provision features directly from Eclipse update sites to the device.

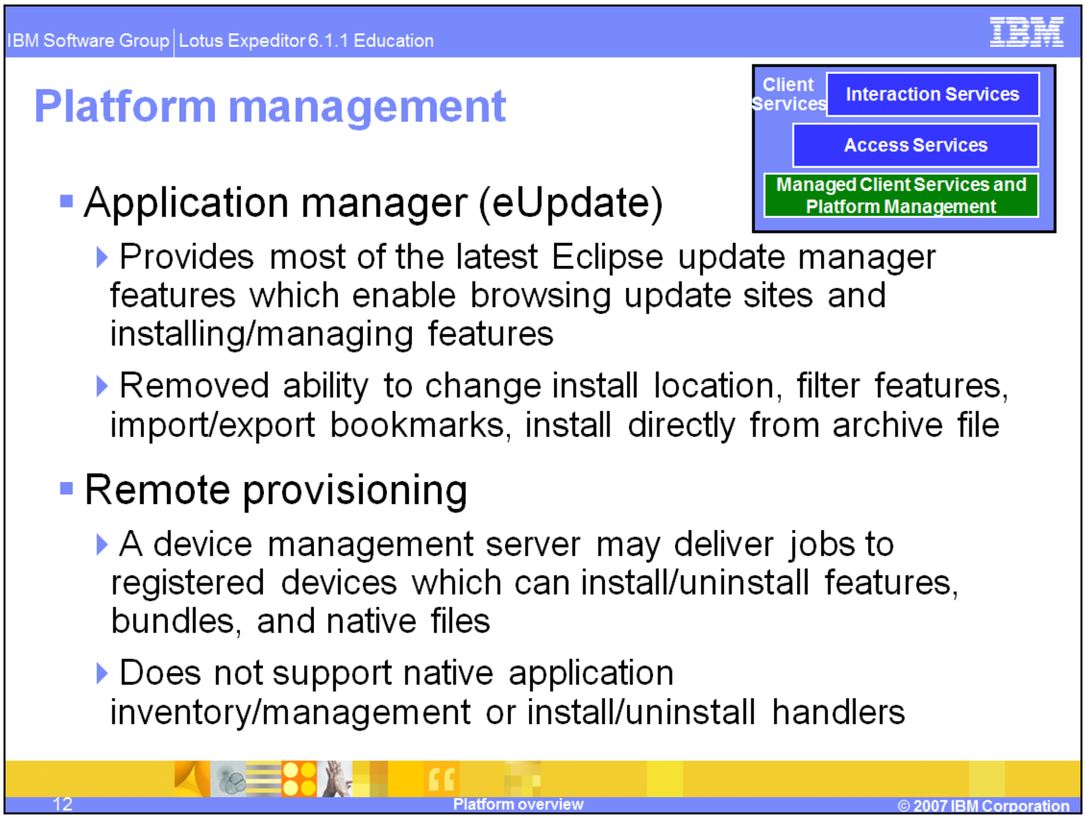


eRCP

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- eRCP is an embedded version of the Eclipse Rich Client Platform
- Utilizes RCP application framework model
- Reduces RCP size/function to fit on devices (~3 MB)
- Pushed changes back to core components to enable running those components on JME CDC/Foundation Profile
- Adds components to enable application binary compatibility across a range of devices
- MicroXML parser – very small/fast (SAX version 2.0, DOM Level 2, and JAXP version 1.0)
- eRCP apps are upward compatible to RCP on the desktop

eRCP is the base platform used in the device runtime. eRCP is from the Eclipse open source project and is an embedded version of Eclipse Rich Client Platform. It uses the RCP application framework model, but reduces the size and function to fit on more capable devices. The project makes patches to mainline Eclipse code so that the most basic components are capable of running on a JME Foundation profile. It also adds some components to better enable application binary compatibility across a broad range of devices. In addition, the MicroXML parser provides a very small and fast SAX2, DOM2 parser. And importantly in an enterprise environment, eRCP applications are also upward compatible to RCP running on a desktop.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Platform management

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

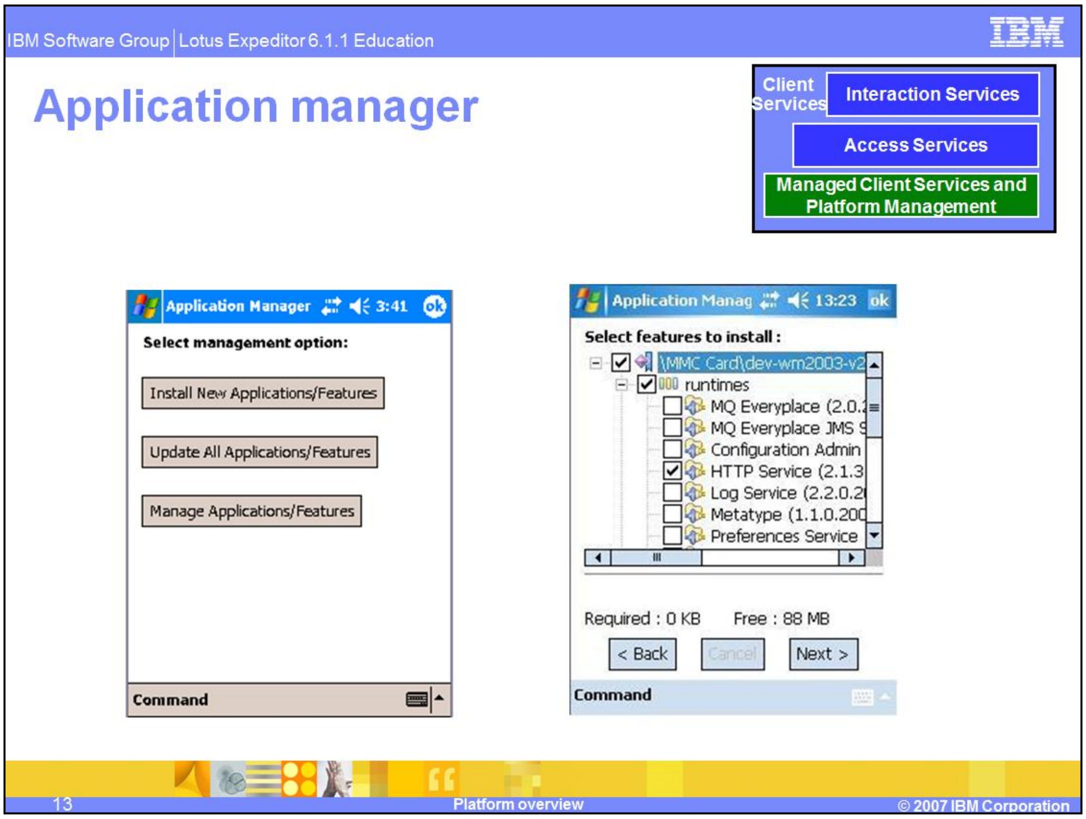
- Application manager (eUpdate)
  - Provides most of the latest Eclipse update manager features which enable browsing update sites and installing/managing features
  - Removed ability to change install location, filter features, import/export bookmarks, install directly from archive file
- Remote provisioning
  - A device management server may deliver jobs to registered devices which can install/uninstall features, bundles, and native files
  - Does not support native application inventory/management or install/uninstall handlers

12 Platform overview © 2007 IBM Corporation

We provide two ways to manage a device platform: Application Manager and remote provisioning.

The Application Manager provides an end-user oriented interface for installing and updating features on the device. Using Application Manager, users can browse update sites, install new features, upgrade existing features, view features details, and remove unnecessary features.

Devices can also be remotely managed by an enterprise administrator. The Enterprise Management Agent and Device Management server provide an efficient way to manage a large number of devices without client interaction. As soon as a device registers to the Device Manager server, the Device Manager server is able to deploy “jobs” to the device to manage it. A system administrator can easily use the Device Manager server to deliver numerous jobs on specific devices, such as to install/uninstall eclipse features, OSGI bundles, or even native files. However, remote provisioning does not support some jobs for native applications that it does for Java applications, such as: inventory collection, native application management, and install/uninstall handlers. Administrators can also arrange for bulk jobs to groups of devices.



Application manager

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

Application Manager 3:41

Select management option:

Install New Applications/Features
Update All Applications/Features
Manage Applications/Features
Command

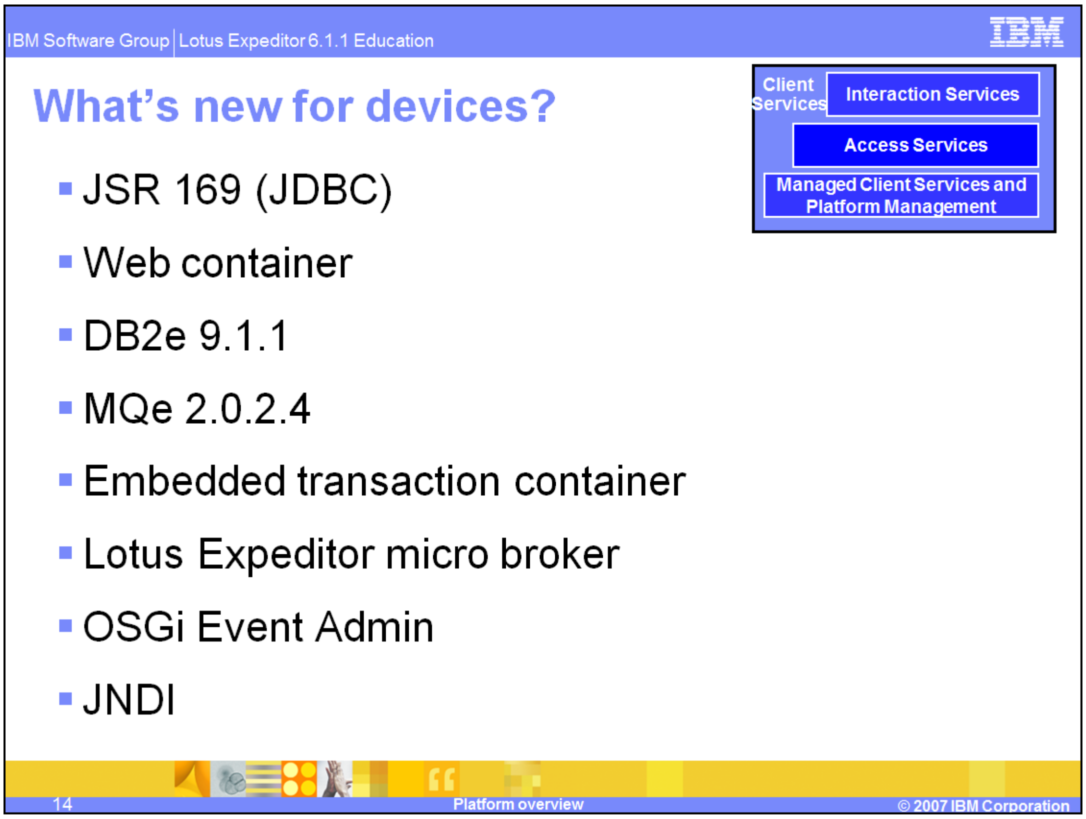
Application Manag 13:23

Select features to install :

\MMC Card\dev-wm2003-v2
runtimes
- [ ] MQ Everyplace (2.0.2
- [ ] MQ Everyplace JMS S
- [ ] Configuration Admin
- [x] HTTP Service (2.1.3
- [ ] Log Service (2.2.0.2
- [ ] Metatype (1.1.0.200
- [ ] Preferences Service

Required : 0 KB Free : 88 MB

This slide displays some screen captures of the Application Manager. The main screen on the left lists the basic functions. The selection screen on the right shows the contents of an update site and allows users to select features and see related information.


IBM Software Group | Lotus Expeditor 6.1.1 Education

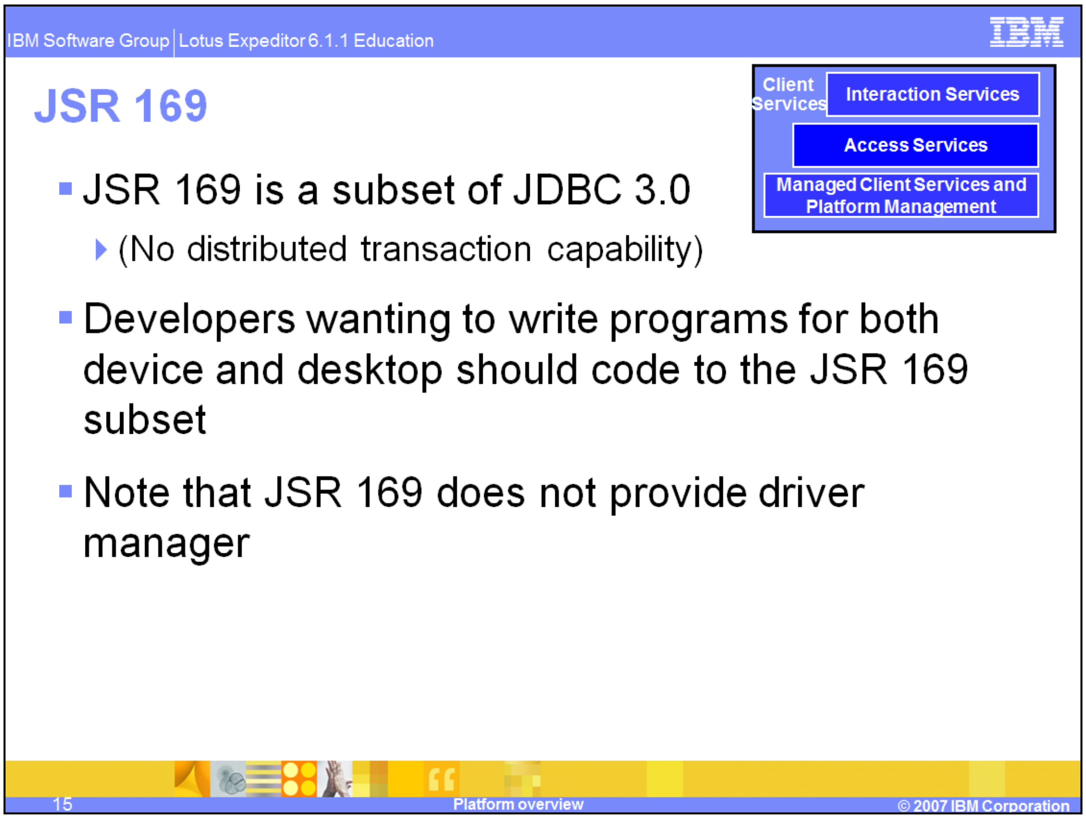
What's new for devices?

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- JSR 169 (JDBC)
- Web container
- DB2e 9.1.1
- MQe 2.0.2.4
- Embedded transaction container
- Lotus Expeditor micro broker
- OSGi Event Admin
- JNDI

14 Platform overview © 2007 IBM Corporation

Some of these components existed in the previous version but are now supported at a higher version. JSR 169, the Embedded Transaction Container, and JNDI are brand new for Lotus Expeditor.

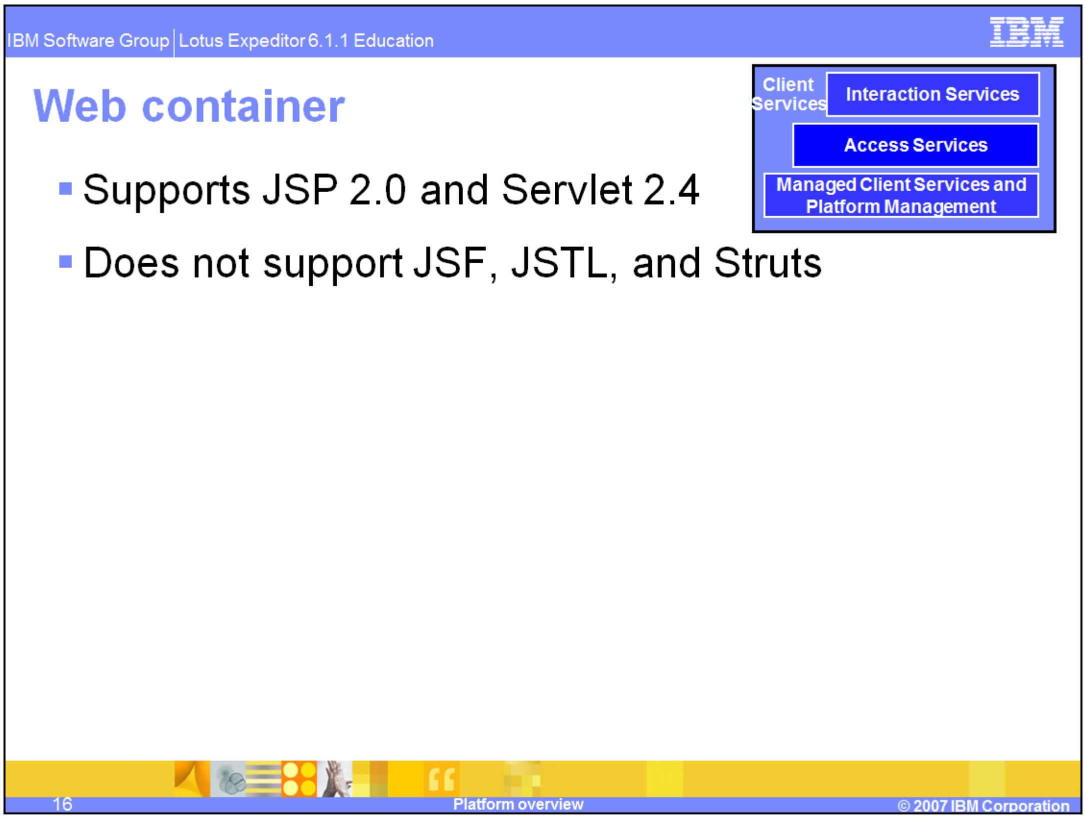


JSR 169

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- JSR 169 is a subset of JDBC 3.0
  - (No distributed transaction capability)
- Developers wanting to write programs for both device and desktop should code to the JSR 169 subset
- Note that JSR 169 does not provide driver manager

JSR 169 does not provide Driver Manager, which could result in some trouble for legacy JDBC 2.0 applications. Developers need to rewrite these JDBC 2.0 segments to use Data source instead.

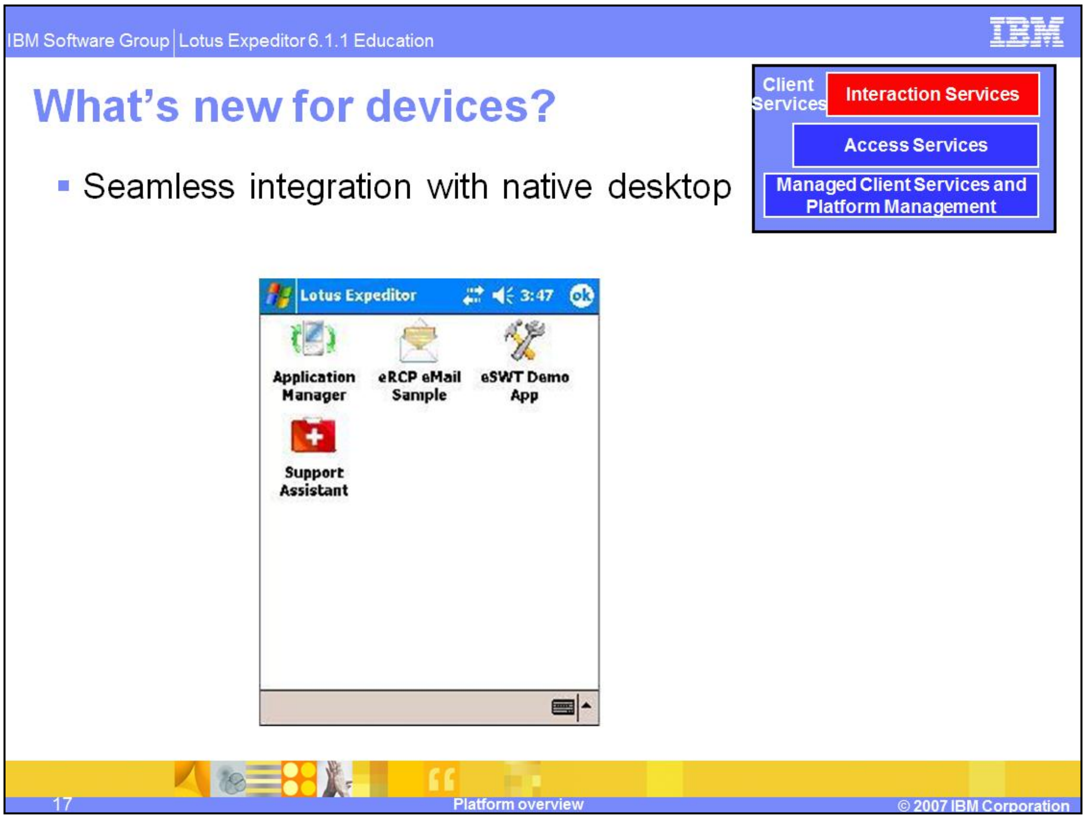


Web container

- Supports JSP 2.0 and Servlet 2.4
- Does not support JSF, JSTL, and Struts

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

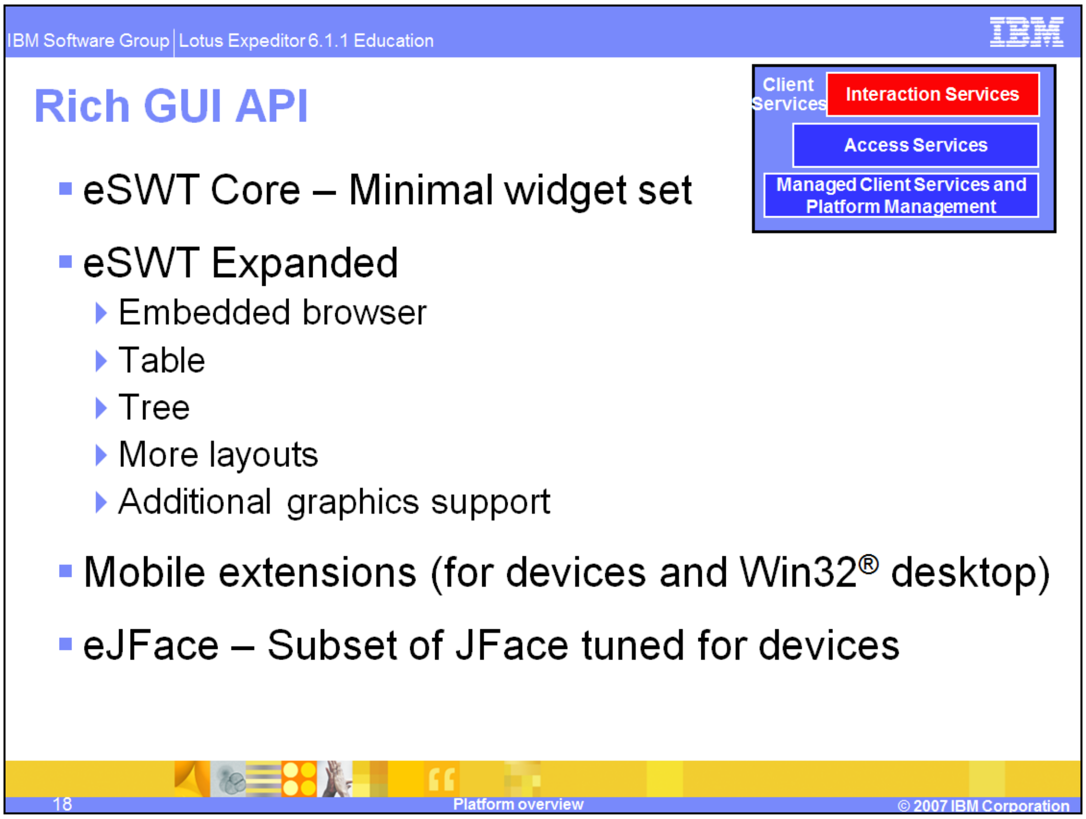
Lotus Expeditor provides two different Web server models: the basic Web Server model is the Http Service, which is an implementation of the OSGi specification for Http Service, and the advanced Web Server model, which is the Web Container. The Http Service model implements an HTTP 1.0 Web server with a Java Servlet 2.1 engine. The Http Service model is recommended if you only need basic http service and do not need the advanced Web Container. The Web Container implements the JSP 1.2 and Servlet 2.3 engines, and it also provides support for configuring multiple HTTP and HTTPS transport channels.



What's new for devices?

- Seamless integration with native desktop

This slide illustrates how eRCP applications seamlessly integrate with the native desktop. Shortcuts to eRCP applications are created and placed under the **Programs** -> **Lotus Expeditor** folder. Users can directly launch each application by simply tapping the icon instead of having to launch a workbench and then choosing what to launch from the workbench.
Applications will also appear in the running programs list like native applications.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Rich GUI API

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- eSWT Core – Minimal widget set
- eSWT Expanded
  - Embedded browser
  - Table
  - Tree
  - More layouts
  - Additional graphics support
- Mobile extensions (for devices and Win32® desktop)
- eJFace – Subset of JFace tuned for devices

18 Platform overview © 2007 IBM Corporation

Here we will introduce the GUI that supports eRCP applications on devices. eSWT stands for embedded Standard Widget Toolkit. It implements a subset of APIs from SWT. Some APIs were removed to make the eRCP size more reasonable for devices.

eSWT Core is the minimum subset of SWT that a device must provide. It contains fundamental user interface elements, including low-level graphics, events, and basic widget infrastructure.

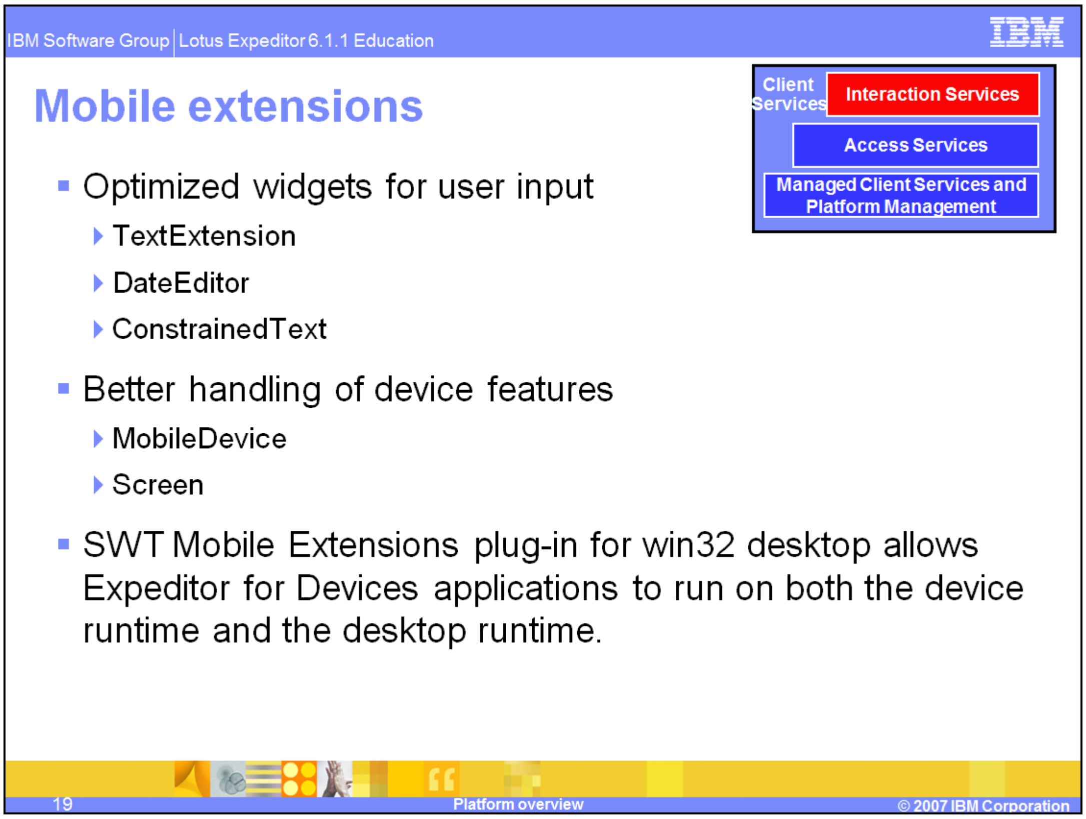
The eSWT Expanded library is also a subset of the SWT. Its purpose is to provide more advanced SWT widgets and layouts that include:

- An Embedded browser that allows the user to visualize and navigate through HTML documents inside an eRCP application.
- A Table widget to provide fancier presentation of the items in a list.
- A Tree widget that shows the structure or hierarchy of related items.
- More layouts to provide optimal use of the screen space.
- Additional image support by the Imageloader in the Expanded library, such as JPEG, GIF, and PNG files.

The widgets in the eSWT Expanded library may be optionally implemented on different devices. eRCP applications should check if these widgets can be instantiated before attempting to use them.

Mobile Extensions are not a subset of SWT. This package implements widgets to enhance the user experience on mobile devices. It also provides functions that abstract varied device hardware differences so that programming an application that can run on multiple devices is not difficult. This is covered on the next slide. The entire Mobile Extensions package is optionally provided, but it is likely to be present on all mobile devices. All eSWT and Mobile Extensions widgets are available on Windows Mobile devices.

eJFace is a subset of JFace. It wraps eSWT widgets, providing a set of components and help utilities to simplify the development of eSWT-based applications.



Mobile extensions

Client Services
Interaction Services
Access Services
Managed Client Services and Platform Management

- Optimized widgets for user input
  - TextExtension
  - DateEditor
  - ConstrainedText
- Better handling of device features
  - MobileDevice
  - Screen
- SWT Mobile Extensions plug-in for win32 desktop allows Expeditor for Devices applications to run on both the device runtime and the desktop runtime.

Let's briefly discuss the Mobile Extensions package in a bit more detail, since it is an essential part of creating generic applications.

Mobile Extension includes optimized widgets for an improved user input experience. This includes the Text Extension widget which records previously typed text that the user has specified. When the user wishes to input the same string, the Text Extension provides completion text; saving the user from typing every character again.

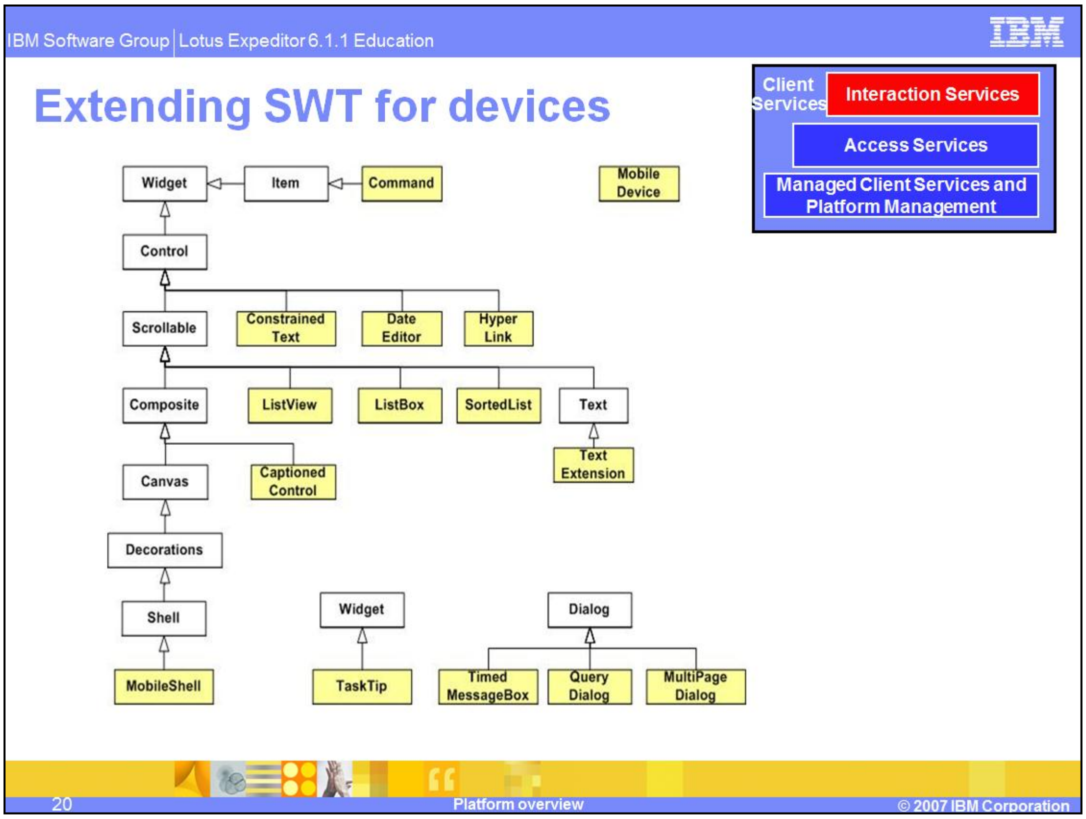
The Date Editor provides many different formats of the Date-Time combination and a calendar for the user to directly pick a date instead of having to type it in. The default format of the Date Editor changes with the locale of the mobile device.

The Constrained Text widget enables the programmer to put a constraint on a Text field, such as URL, Phone number, and E-mail. This prevents the user from inputting the wrong format or incorrect information.

For better handling of device features we have the Mobile Device class which enables application programmers to manipulate the behavior of the Virtual keyboard on the device. It is reasonable to set the virtual keyboard to always be off for a mobile device that already has a keyboard in order to save precious screen space for other widgets.

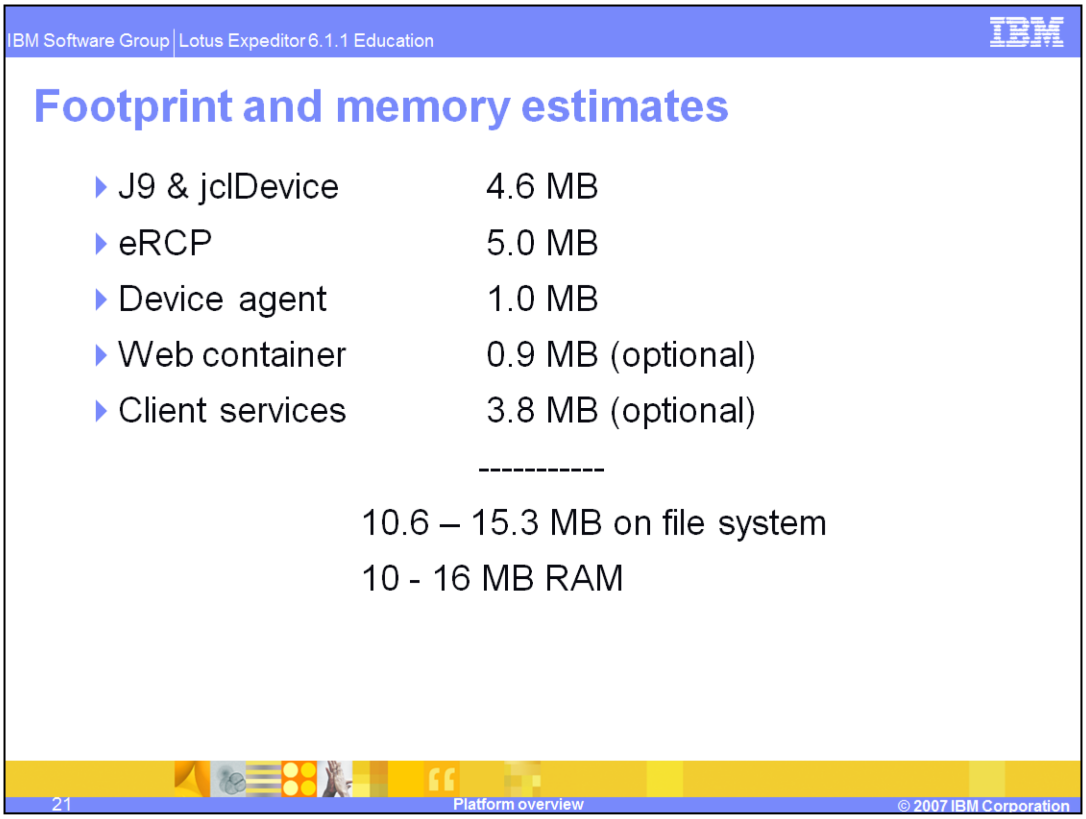
The Screen class provides useful information about each screen in the mobile device. The information includes the size of the screen, the orientation of the screen, and whether the screen is a touch screen. The programmer may decide which widgets are better to use for an application depending on the constraints of each screen.

The SWT Mobile Extensions plug-in for win32 desktops allows Expeditor for Devices applications to run on both the Device runtime and the Desktop runtime.



Extending SWT for devices

This slide shows the hierarchy of basic SWT widgets and how Mobile Extensions widgets extend it. The widgets shaded in yellow are the Mobile Extension widgets.



Footprint and memory estimates

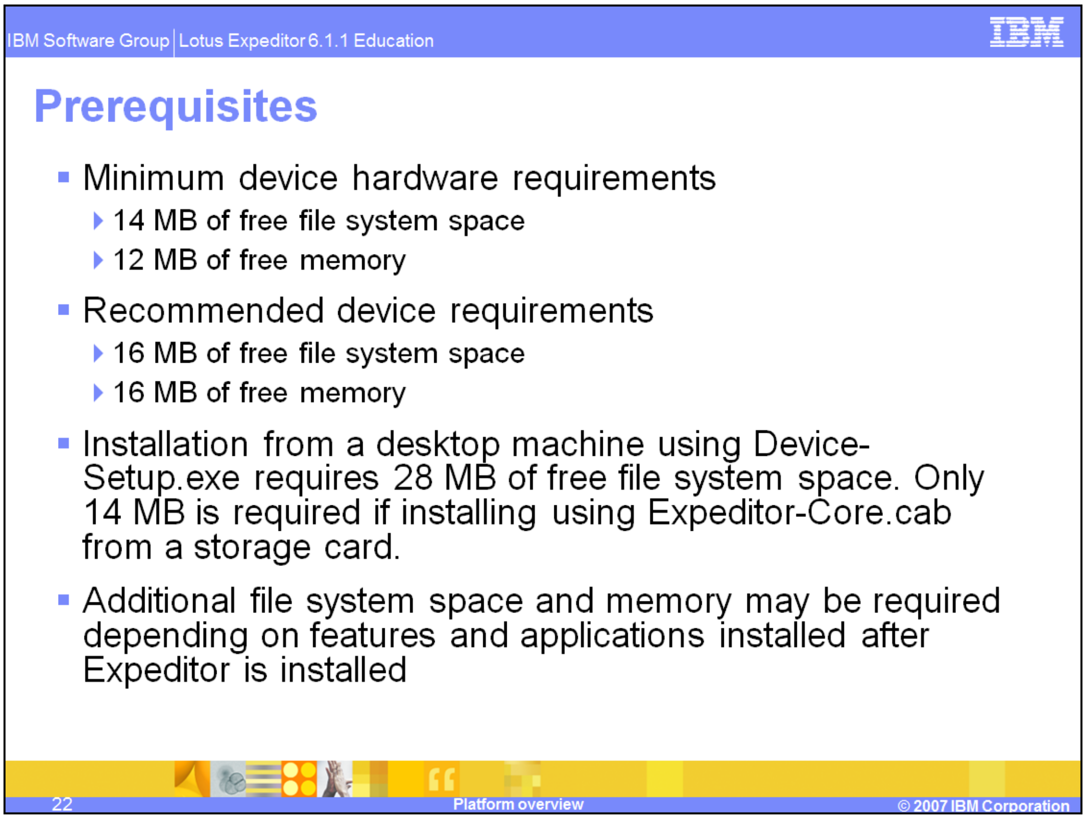
J9 & jclDevice	4.6 MB
eRCP	5.0 MB
Device agent	1.0 MB
Web container	0.9 MB (optional)
Client services	3.8 MB (optional)

10.6 – 15.3 MB on file system

10 - 16 MB RAM

On a Windows Mobile device, J9 and jclDevice require around 4.6 MB. eRCP 1.0 requires around 5 MB. Device Agent needs 1 MB. The optional Web Container needs 0.9 MB and other optional Client Services require 3.8 MB. To sum up, the Expeditor footprint ranges from 11 to 16 MB. Note that most Windows Mobile devices use a compressed file system and so actual flash consumption may be considerably less.

The RAM requirements range from 10 to 16 MB depending on how many components are actively running and what languages are installed. Additional resources are required for applications.


IBM Software Group | Lotus Expeditor 6.1.1 Education

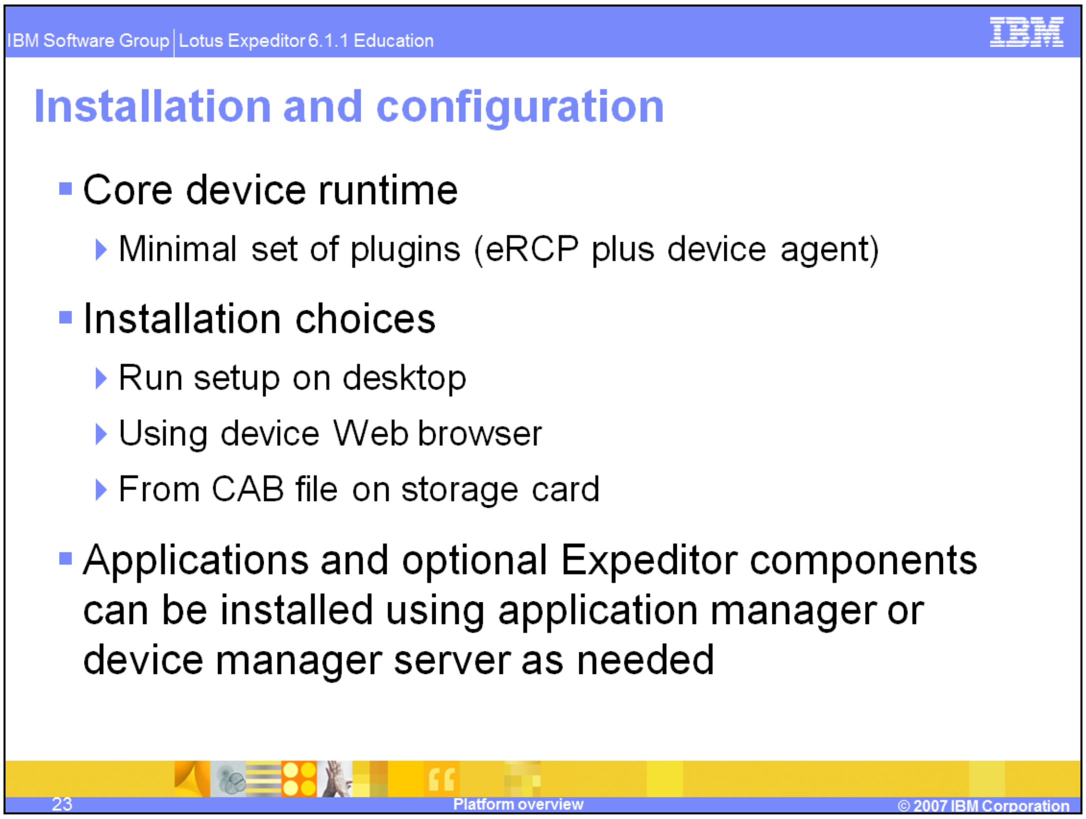
Prerequisites

- Minimum device hardware requirements
  - 14 MB of free file system space
  - 12 MB of free memory
- Recommended device requirements
  - 16 MB of free file system space
  - 16 MB of free memory
- Installation from a desktop machine using Device-Setup.exe requires 28 MB of free file system space. Only 14 MB is required if installing using Expeditor-Core.cab from a storage card.
- Additional file system space and memory may be required depending on features and applications installed after Expeditor is installed

22 Platform overview © 2007 IBM Corporation

To run Lotus Expeditor Client on your Windows Mobile device, you need to have at least 12 MB of free file system space, 12MB free memory and 10 MB free virtual address space. However, to install more features and applications after Expeditor is installed requires additional file system space and memory.

Note that there are significant limitations in the Windows Mobile memory architecture which may restrict the execution of large applications. Windows Mobile has a per process memory usage limit which can be reached even though there is more than enough memory available within the device. This limitation is more likely to be encountered on Asian language devices since they use more memory resources.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Installation and configuration

- Core device runtime
  - Minimal set of plugins (eRCP plus device agent)
- Installation choices
  - Run setup on desktop
  - Using device Web browser
  - From CAB file on storage card
- Applications and optional Expeditor components can be installed using application manager or device manager server as needed

23 Platform overview © 2007 IBM Corporation

Since some devices are not capable of running the entire Expeditor for Devices runtime, you initially install only a "core" runtime, which contains only the minimal set services. The rest of Expeditor is provided as optional features in an update site. The core runtime specifically contains the Eclipse eRCP runtime, the Enterprise Management Agent, Application Manager, and Support Assistant.

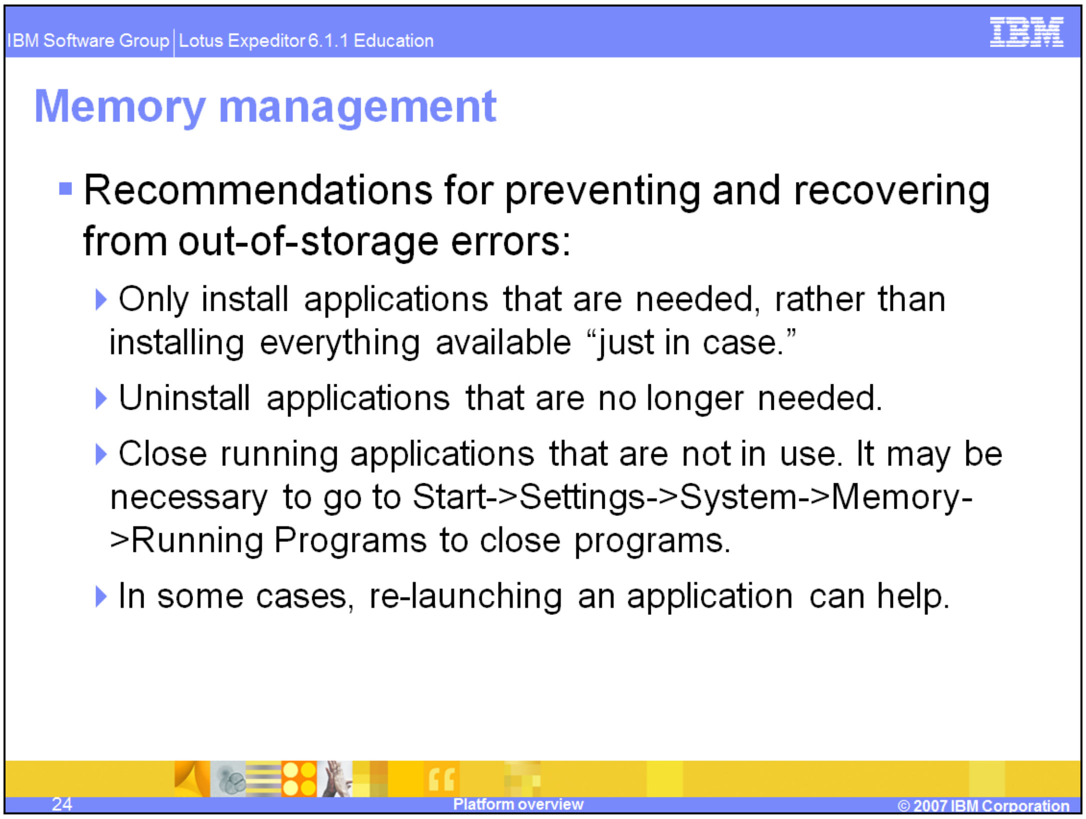
You have several choices for installing the Expeditor core runtime:

- You can run setup on the desktop. This installs Expeditor on a connected device. This is best done when there is a one-to-one relationship between the desktop machine and the device.

- You can open a browser on the device and browse to the installation cab file, then click on it to start the installation.

- You can also copy the cab file to a storage card, then click on it from File Explorer.

For other applications and optional Expeditor components, you can either install using Application Manager or the Device Manager system. This depends on what your strategy is for deploying applications.


IBM Software Group | Lotus Expeditor 6.1.1 Education

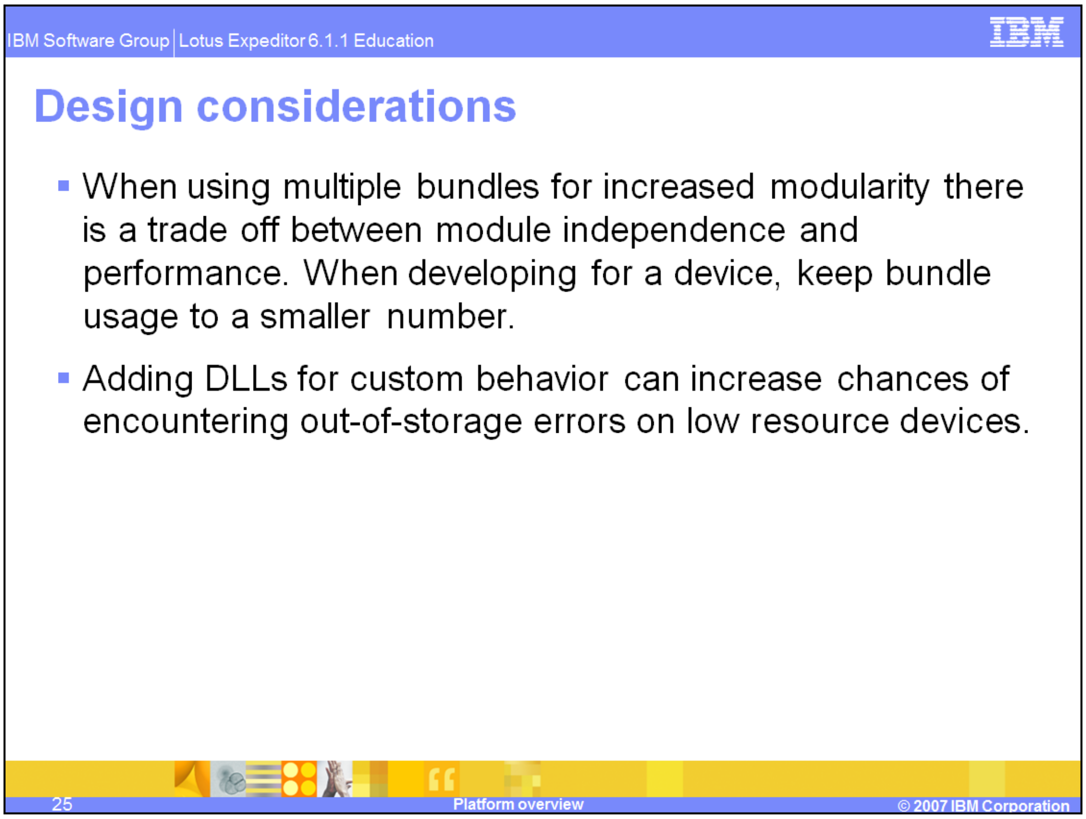
Memory management

- Recommendations for preventing and recovering from out-of-storage errors:
  - Only install applications that are needed, rather than installing everything available “just in case.”
  - Uninstall applications that are no longer needed.
  - Close running applications that are not in use. It may be necessary to go to Start->Settings->System->Memory->Running Programs to close programs.
  - In some cases, re-launching an application can help.

24 Platform overview © 2007 IBM Corporation

Out of memory conditions can happen more frequently on devices than on desktops. Here are some hints for how to manage your device’s memory.

- Don’t install everything you think you may someday need. Only install the applications you need today.
- When you no longer need an application, uninstall it.
- You can close running applications not currently being used. To close them, go to **Start** > **Settings** > **System** > **Memory** > **Running Programs**.
- Sometimes, re-launching an application may work if it doesn’t start on the first try. This is because on an “out of memory” condition. The operating system may ask other applications to close, thereby freeing additional resources.



Design considerations

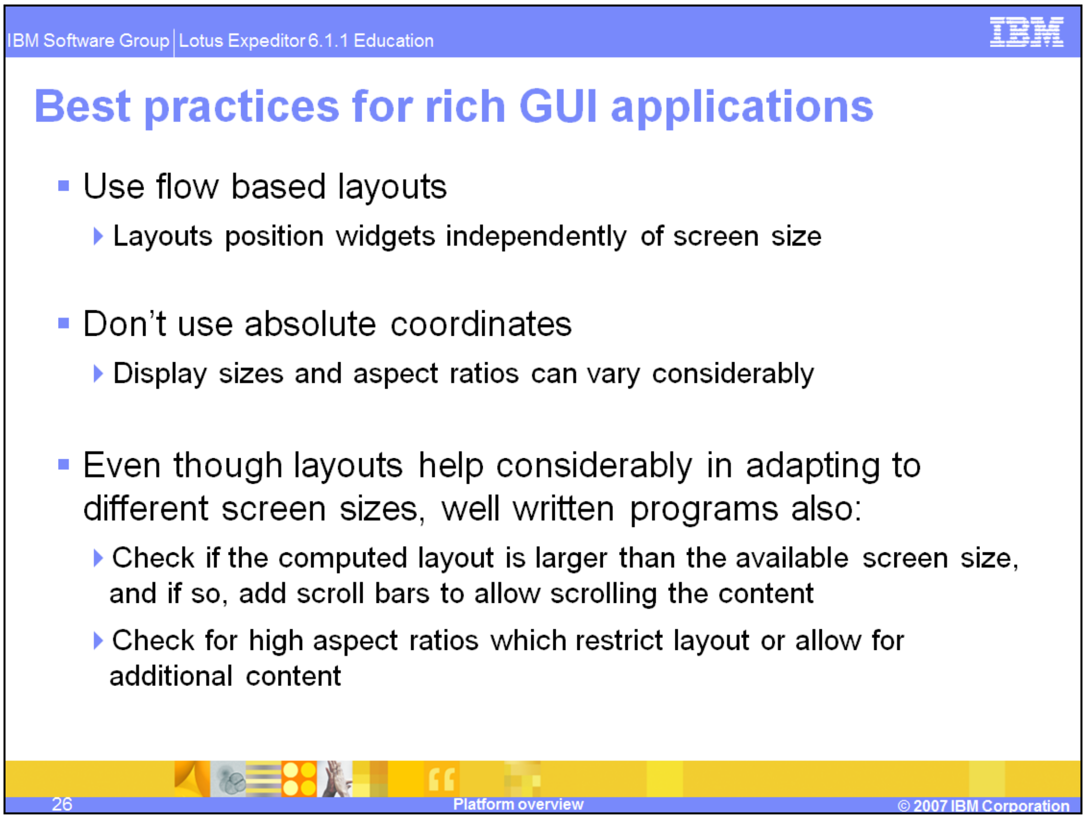
- When using multiple bundles for increased modularity there is a trade off between module independence and performance. When developing for a device, keep bundle usage to a smaller number.
- Adding DLLs for custom behavior can increase chances of encountering out-of-storage errors on low resource devices.

Designing an application for devices is a bit different than designing one for desktop systems.

You might want to use multiple bundles to increase modularity; however, the more bundles, the more drag on startup performance. Therefore, it is best to keep the number of bundles limited.

If you add DLLs onto a device that does not have good virtual memory resources, you are likely to get “Out of storage” errors.

For logging, it is recommended you use OSGi logging. If you have legacy code that uses Java Logging, you may still use this code for now and change it later.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Best practices for rich GUI applications

- Use flow based layouts
  - Layouts position widgets independently of screen size
- Don't use absolute coordinates
  - Display sizes and aspect ratios can vary considerably
- Even though layouts help considerably in adapting to different screen sizes, well written programs also:
  - Check if the computed layout is larger than the available screen size, and if so, add scroll bars to allow scrolling the content
  - Check for high aspect ratios which restrict layout or allow for additional content

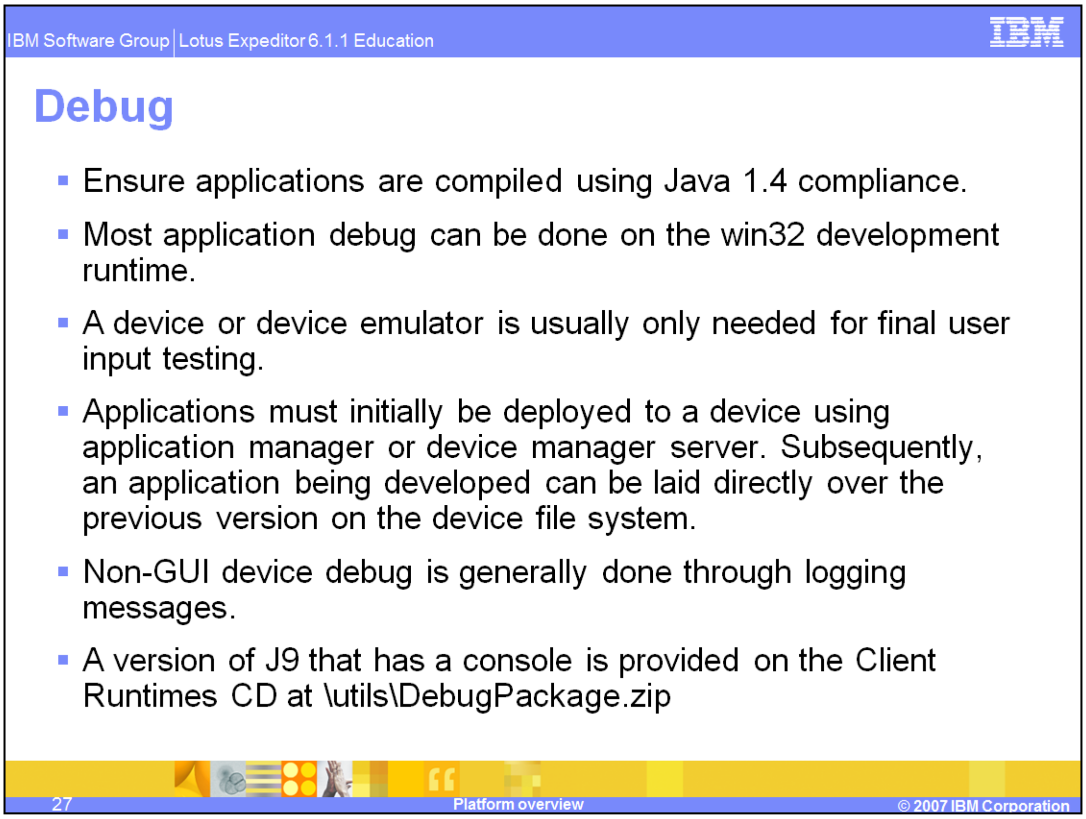
26 Platform overview © 2007 IBM Corporation

Here are best practices for designing Rich GUI applications:

It is best to use flow-based layouts. Layouts automatically position each widget using the available screen space. The layout will re-position each widget in case of a change in screen size or screen orientation to best display each widget and avoid having them clipped.

Don't position widgets using absolute coordinates. There is a considerable chance that moving the same application from device to device will cause widgets to be clipped or not even show up on the screen.

Even though layouts help considerably in adapting the user interface to different screen sizes, we also suggest you check if the available screen size is large enough to contain the computed layout. If it is not, then scroll bars should be added to make all the widgets available for the user. We also encourage checking for aspect ratios which restrict layouts or allow for additional contents to be displayed.


IBM Software Group | Lotus Expeditor 6.1.1 Education

Debug

- Ensure applications are compiled using Java 1.4 compliance.
- Most application debug can be done on the win32 development runtime.
- A device or device emulator is usually only needed for final user input testing.
- Applications must initially be deployed to a device using application manager or device manager server. Subsequently, an application being developed can be laid directly over the previous version on the device file system.
- Non-GUI device debug is generally done through logging messages.
- A version of J9 that has a console is provided on the Client Runtimes CD at \utils\DebugPackage.zip

27 Platform overview © 2007 IBM Corporation

This slide describes some debug issues you might encounter.

Firstly, make sure your applications are compiled against Java 1.4. Java 5.0 compiled code will work in the Expeditor runtime.

We also provide the same Expeditor device runtime for Win32 development platform; most debug can be done there.

If you want to test user input and how the device user interface will look on the device, then you can either deploy the application to the device or try a device emulator. To deploy your applications, you need to go through either application manager or Device Manager server. As a shortcut, if you have an older version application already installed on the device, you can simply overlay the plug-in with the newer one.

For a non-GUI application, logging is a good way to do debug.

You can get a console version of J9 from \utils\DebugPackage.zip on the Client Runtimes CD. This will allow you to access the OSGi console and manipulate plug-ins directly.

Trademarks, copyrights, and disclaimers