

# How to Develop eRCP Workbench Applications

## Target Audience:

Mid-range and High-end mobile device application developers

## About this Guide:

This guide describes the development process for the eRCP 1.0 GA build. eRCP supports two different application models, stand-alone and workbench plug-in. Prior to the eRCP M4 build, only the stand-alone application model was supported. The document “How to Develop eRCP Apps on Eclipse 3.1” describes development of stand-alone applications. This document discusses how to develop, install, and run applications which are independent workbench plug-ins.

## Development Environment:

Windows 2000, XP  
Java 1.4.x or higher  
Eclipse 3.2

## Target Environment:

Mobile Device capable of running J2ME – CDC/Foundation Profile  
3MB of available storage  
6MB of available RAM

## Information about eRCP:

[www.eclipse.org/ercp](http://www.eclipse.org/ercp)

## Overview:

The primary difference between the stand-alone model and workbench plug-in model is that the stand-alone model allows only one GUI application per JVM and that app owns its entire display window. This is similar to the traditional RCP application model. In the workbench plug-in model, applications run simultaneously in a single workbench window which controls where and when applications are displayed. The workbench application model is closely aligned with the standard Eclipse IDE Workbench model in which various tools plug into the Eclipse IDE. Workbench plug-ins generally provide one or more views. One difference between IDE plug-ins and eRCP plug-ins is that eRCP plug-ins do not provide perspectives.

## The basic parts of an eRCP Workbench application:

### UIPlugin class:

The plug-in must have a class which extends `org.eclipse.ui.plugin.AbstractUIPlugin`. This provides the base functionality for a starting and stopping the application within the OSGI framework.

### View class(es):

The plug-in must have at least one class which extends `org.eclipse.ui.part.ViewPart`. These classes provide the GUI implementation for the application. eRCP applications may provide several Views which may be called upon by the workbench for displaying the application appropriately on varying device hardware. The View types are Normal, Large, and Status. A Normal View is required. The other two are optional. The Large View is for VGA or higher resolution screens. The Status View is for very small screens which have room only for very limited status information.

A View class must implement the `createPartControl()` method. This is where the application initially creates its GUI on an eSWT Composite widget.

A View class must implement the `setFocus()` method in order to give focus to an appropriate control when the View as a whole gains focus.

**Manifest:**

The manifest file is an XML file but is best viewed using the Manifest Editor in Eclipse.

Under the Overview tab within the editor, the **ID** field is normally the package name of the plug-in. The **Version** field is set as you like. **Name** is a human readable name for the application. **Provider** is optionally your name or company name. **Class** must be set to the fully qualified class name of your class which extends AbstractUIPlugin.

Under the Extensions tab within the editor, org.eclipse.ercp.eworkbench.applications and org.eclipse.ui.views extensions are required.

For the **org.eclipse.ercp.eworkbench.applications** extension, the properties of “application” must be set. An application **id** is required and is typically the application package name. **Name** is a human readable description. **Singleton** specifies whether the workbench can launch multiple instances of the application or only one. For the properties of “views”, at least the **normal** field must be set. This is the reference id of a specific view (see next paragraph).

For the **org.eclipse.ercp.eworkbench.views** extension, the properties of “view” must be set. A view **id** is usually defined as the combination of the view’s package name with a “.normal” postfix. **Name** is a human readable description. **Class** specifies the particular View class (from above) that implements the view. **Category** must be one of:

org.eclipse.ercp.category.normal  
org.eclipse.ercp.category.large  
org.eclipse.ercp.category.status

**Icon** specifies an icon which represents the application and is scaled for the appropriate view. **FastViewWidthRatio** is ignored. **AllowMultiple** determines whether multiple views may be instantiated.

**The eWorkbench:**

One of the eRCP project components is the “Generic eWorkbench”. This stand-alone application owns the JVM’s GUI thread and manages the launch and display of all eRCP workbench applications. It is called “generic” because it is a simple workbench that is meant to run on any hardware. It can be used as the base for more advanced workbenches which take advantage of particular hardware features. For instance, a mobile phone with multiple displays may need to display an application’s *status* view on the front of the phone and the *normal* view on a larger display when a cover is opened.

The generic eWorkbench initially displays a list of available eRCP plug-ins. When the end user selects one, its normal view is displayed. A Command is available to switch back to the application list.

**Installation on Windows development machine:**

Download the Eclipse 3.2 SDK – [www.eclipse.org/downloads](http://www.eclipse.org/downloads)

Download the eRCP 1.0 package for Windows – [www.eclipse.org/ercp](http://www.eclipse.org/ercp)

Install Eclipse by unzipping the download file

Unzip eRCP 1.0 package

**Note: the following installation item supports embedded development and is not necessary for running eRCP applications on desktop machines.**

Download WSDD 5.7.1 – [www.ibm.com/embedded](http://www.ibm.com/embedded)

Install WSDD

### **Run Eclipse on development machine:**

Run %EclipseHome%\eclipse.exe

In order to build and run against the correct libraries, point target platform at eRCP directory

Click Window, then Preferences

Expand Plug-in Development and click Target Platform

Enter location of eRCP (i.e. eRCP-v20060921-1641\win32\eRCP) and click OK

Change Java compiler version to be compatible with Foundation JVMs

Click Window, then Preferences

Expand Java and select Compiler

Change the Compiler compliance level to "1.4"

Import org.eclipse.ercp.app sample into workspace

Click File, then Import

Select Checkout projects from CVS

Select Create a new repository location and fill in following information:

Server: dev.eclipse.org

Path: /home/dsdp

User: anonymous

Choose Use specified module name: org.eclipse.ercp/org.eclipse.ercp.app

Click Finish

Switch to plug-in perspective (via tab on upper right of IDE)

**Note: the following 3 sections support embedded development and are not necessary for running eRCP applications on desktop machines.**

### **Setup workspace for development on Foundation Profile:**

Click Window->Preferences, then select Java->Installed JREs

Add a new JRE

Choose "Standard VM"

Provide an appropriate name such as "Foundation"

For JRE home, point to a standard J2SE JVM

Uncheck "Use default system libraries"

Delete all libraries listed

Add new libraries that point to your desired configuration. I.e.:

%WSDHome%\wsdd5.0\ive-2.2\lib\charconv.zip

%WSDHome%\wsdd5.0\ive-2.2\lib\jclFoundation10\classes.zip

%WSDHome%\wsdd5.0\ive-2.2\lib\jclFoundation10\locale.zip

%WSDHome%\wsdd5.0\ive-2.2\lib\jclFoundation10\map.zip

Note: Setting the default JRE affects the entire workspace except for those projects that override the default with their own preference. You can set the workspace default to Foundation and still keep your "Run..." configurations as J2SE.

### **Set sample project to use Foundation JRE**

Right click on org.eclipse.ercp.app project

Select Properties and Java Build Path

Select Libraries tab

Remove JRE System Library

Click Add Library and select JRE System Library

Check Alternate JRE, and select Foundation JRE, then click Finish

### **Running sample on desktop machine:**

Click on Run -> Run...

Select Eclipse Application and click New icon

Enter Name of eWorkbench

Select Plug-ins tab and check "Launch with all workspace and enabled external plug-ins"

Click Run

eWorkbench should appear in a new window and display eRCP App as a selectable application

Click on eRCP App to switch to the application's view

Select App List Command to return to the list of applications

### **Sample Description:**

The sample demonstrates an application provided as an OSGi bundle. It is a workbench plug-in which provides a view that can be displayed by the workbench. The view utilizes eJFace's TableViewer widget to display a table of items with associated icons.

### **AppPlugin.java Code:**

This class is required to fit the sample into the eRCP application model. It extends AbstractUIPlugin which lets it be discovered and loaded by the OSGi runtime. Methods in the class allow the runtime to start and stop the plug-in.

### **NormalView.java Code:**

This class implements the application GUI by making eSWT calls to display the application's UI on a Composite provided by the workbench. The workbench calls CreatePartControl() when the application's view needs to be presented for the first time. Internal classes within this class act as providers for a TableViewer widget.

### **LargeView.java Code:**

This class implements the application GUI to be used on large displays. Note that in this example this class is largely a copy of the NormalView class. It would probably be more efficient to share more of the GUI code, rather than copy it.

### **SamplePage.java and SampleSubPage.java Code:**

These classes implement the GUI for preference pages that are specified in the Extensions tab of the manifest.

### **Manifest Description:**

The manifest file records information about the plug-in needed for it to be found and executed. For convenience, I'll refer to sections of the manifest as shown by the Eclipse Manifest Editor.

Under Overview, we specify the full plug-in class name of org.eclipse.ercp.app.

Under Dependencies, the following plug-ins are **Required Plug-ins:**

org.eclipse.core.runtime

The following packages are **Imported Packages:**

org.eclipse.ercp.swt.mobile

org.eclipse.jface.action

org.eclipse.jface.preference

org.eclipse.jface.resource

org.eclipse.jface.viewers

org.eclipse.swt

org.eclipse.swt.events

org.eclipse.swt.graphics

org.eclipse.swt.layout  
org.eclipse.swt.widgets  
org.eclipse.ui  
org.eclipse.ui.dialogs  
org.eclipse.ui.part  
org.eclipse.ui.plugin

These packages are specified by *package name* versus *plug-in ID* because the deployment of these packages may be done through a variety of plug-in configurations. For instance, in the M3 build, Core eSWT, Expanded eSWT and SWT Mobile Extensions were packaged as three separate fragments. However, in the M4 build, they are all packaged together in a single fragment. This method of satisfying dependencies also permits upward compatibility with machines running desktop SWT.

Under Extensions, org.eclipse.core..runtime.applications and org.eclipse.ui.views are specified as described above under “**The basic parts of an eRCP Workbench application**” section. The org.eclipse.ui.preferencePages extensions define two preferences pages. These pages will be available under the “Preferences” command when the sample is the foreground application in the workbench.

Under Build, Runtime Information, a dot path specifies that classes can be found by searching from the root of the plug-in. The items checked under Binary Build determine what else will be included when the sample plug-in is built.

The remaining tabs show the actual XML built from information provided in the previous tabs and usually does not need to be altered.

#### **Deploying eRCP to a Windows Mobile target device:**

Download the eRCP 1.0 package for WM2003 – [www.eclipse.org/ercp](http://www.eclipse.org/ercp)

Unzip the downloaded file

Follow the platform read.me for installation instructions (you may also need to install a JVM)

Run eWorkbench by launching file explorer and clicking on j9foun-eworkbench

## **Writing your own eRCP Workbench Application**

#### **Checkout sample project:**

If you have not already done so, use CVS Repository Explorer to check-out

org.eclipse.ercp/org.eclipse.ercp.app

Right click on project and choose Team/Disconnect

Choose to delete CVS data

#### **Rename and modify sample project:**

Right click on project

Choose Refactor/Rename and enter a new project name

Modify the sample code as you like

Modify the manifest to uniquely identify your new plug-in application. There are several IDs to be modified. If you do not make unique IDs, the your app will clash with the existing eRCP sample app.

#### **Test sample in IDE:**

Run the eWorkbench as an Eclipse Application (see above)

Your application should appear in the application list and be launchable

**Exporting the sample plug-in:**

Right click on org.eclipse.ercp.app project  
Click on Export  
Select Deployable plug-ins and fragments and click Next  
Select the org.eclipse.ercp.app project  
Under Export Destination, select Directory and choose eRCP-v20060921-1649\wm2003\eRCP or your own runtime directory  
Click Next and then Finish  
The plug-in is now deployable to a device

**Running sample on a device:**

Copy the sample application jar from the eRCP\plugins directory to the same directory on your target device  
Remove the folders under eRCP\configuration. Do not remove the config.ini file. (This removes the OSGi cache. Without this step, your plugin will not be seen by the OSGi runtime.)  
Run the sample app you built by launching file explorer and clicking on j9foun-eworkbench