Extension Services

Platform Builder

Contents

Chapter 1. Tasks

Creating a Platform Builder project

You can create multiple Platform Builder projects for different target device configurations or contents. To create a new Plaform Builder project, select **File > New > Project > Extension Services > Platform Builder** to begin the Platform Builder wizard process. The wizard will create a new Platform Builder project. You can use this project to build target platforms. You can further customize it by using the Platform Builder Properties Editor.

New Project page

Input the project name and select a new project path if the default path is undesirable. If the **Use Default** checkbox is not checked, provide the full path of the project.

Platform Options page

The Platform Options page defines the primary characteristics of the Platform Builder project. This page contains sections dealing with the project's device, JVM, and build output options.

The Device group contains the following controls:

- **Type**

  The type of device that the target platform will execute on. Five predefined types are available that make use of the J9 JVM. A Custom type is also available that requires the inclusion of a custom JVM. The Custom option is available as a convenience, but target platforms built using the Custom device type are not supported.
- **Operating system**

  The list of OSGi supported operating systems. If you selected a predefined device type, this control will be preselected and disabled. If you selected a Custom device type, this control will be enabled.
- **Processor**

  The list of OSGi supported processors. If you selected a predefined device type, this control will be preselected and disabled. If you selected a Custom device type, this control will be enabled.

The JVM group contains the following controls:

- **Type**

  A label displaying the type of JVM that the target platform will contain. This value is set based on the Device Type specified in the previous group. If a predefined device type is selected, this value will be set to J9 and a J9 JVM will be used. If a custom device type is selected, this value will be set to Custom and a custom JVM location must be provided.
- **Custom JVM location**

  If you selected Custom for the JVM type, specify a filesystem location to the custom JVM. This value should typically point to a filesystem location that contains bin and lib directories.

The Build Output group contains the following controls:

- **Default target root**

  Select this checkbox if you wish to use the default target root location for the platform. The default root for the target platform is "/", meaning that the build output will contain the smf and DVM directories at the base location of the .zip file (or whatever package format is specified).
- **Custom target root**

  If the default target root is not selected, a custom target root should be specified. A custom target root would be needed, for example, if you wanted your target platform to be packaged under the "/opt/platform/" directory.
- **Output format**

  Select the file type you want the target platform build output packaged in. The available package formats are:

  1. **ipk** - The iPKG format typically used on Zaurus devices
  2. **tar** - The standard TAR format typically used on Linux devices
  3. **tar.gz** - A zipped version of the TAR format typically used on Linux devices
  4. **zip** - The ZIP format used for Windows devices
- **Output location**

  The location you enter here is where the Platform Builder places the target platform build output upon a successful project build. If no location is filled in, the default output location will be the output directory of the Platform Builder project.

J9 Options page

If you select J9 as the target JVM, the J9 Options page will display. The page provides the following controls:

- **J9 Class Library**

  The following class libraries are implementations of a subset of the Java 2 class library specifications. All class libraries are based on the Java Development Kit (JDK) 1.3 specification.

Table 1. J9 class libraries

Library	Description/Use
jclRM (~1 MB)	jclRM should be used in situations where a small footprint and a reasonable level of functionality is required.
jclFoundation (~1.5 MB)	jclFoundation is an implementation of the Connected Device Configuration profile that also contains security support.
jammed (~2 MB)	jammed is available for situations where memory utilization is a lesser issue and more advanced functionality is required.

- **Package format**

  The target platform may contain multiple packages of Java classes. These can either be structured as Java archive files (JAR) or as J9 executable (JXE) files. JXE files use most of a specific embedded device's limited resources by paring down the classes to include only what is necessary for execution. The J9 Smartened generates JXE files. JXE files are only supported on a J9 JVM.
- **Optional libraries**

  There are several optional J9 libraries not required to execute an Extension Services platform. However, you may need these libraries for other purposes. Selecting the libraries in this section will include them in the target platform.

Table 2. Optional libraries

Library	Description/Use
Debug	Provides the debug libraries for J9 (for use with the "-debug:<options> -Xrdbginfo:<host>:<port>" J9 command line options)
Just-in-time compiler	Enables the just-in-time (JIT) compiler
Ahead-of-time compiler	Enables ahead-of-time (AOT) compilation of JXE files
Freely distributable LIBM	A C math library for machines that support IEEE 754 floating-point arithmetic. It provides the native functions required by the class java.lang.StrictMath and is, therefore, not needed if this class is not used
Profiling	Provides MicroAnalyzer support on the target platform (for use with the "-analyze" J9 command line option)
Verbose output	Allows for use of the "-verbose[:class,gc,stack,sizes]" J9 command line option

Startup Options page

The Startup Options page contains sections dealing with the target platform's JVM and SMF options.

The JVM Options group contains the following controls:

- **Bootclasspath**

  Text entered here appends to the Java bootclasspath when you start the target platform.
- **Classpath**

  Text entered here appends to the Java classpath when you start the target platform.
- **Properties**

  Text entered here passes to Java when you start the target platform.

The SMF Options group contains the following controls:

- **Console support**

  Selecting this option will include the SMF console in the target platform. You can also provide an optional port to allow for telnet access to the SMF console.
- **Bundle Developer support**

  Selecting this option will include support for connecting SMF Bundle Developer to the target platform. You must specify a port for this option.
- **Resource management support**

  Selecting this option will include resurgence management support for the target platform.
- **Bundle Server address**

  A network address (IP address or hostname) and port entered here will allow the target platform to communicate with an SMF Bundle Server.

Languages page

The Languages page allows for the selection of languages which the target platform will support. Select multiple languages by clicking the **Select...** button and choosing the desired languages from the dialog that appears. At least one language must be selected.

Custom Bundles page

The Custom Bundles page lets you add custom bundles to the target platform.

To add custom bundles to the list, click the **Add...** button and select one or more bundle files. To remove custom bundles from the list, select the bundles and click the **Remove** button.

Click the **Compute** button to examine the current list of target platform bundles and determine what other bundles need to be added to the target platform to resolve any package or service dependencies. A series of dialogs will appear that will denote which bundles should be added to the platform to fulfill dependencies and which bundles should be removed due to conflicting packages. As the dialogs are displayed, you can choose to select bundles and click **Resolve...** to apply changes, or click **Ignore...** and skip over the suggested bundles.

System Bundles page

The System Bundles page lets you add system bundles to the target platform. To add system bundles to the list, click the Add... button and select one or more bundles. To remove system bundles from the list, select the bundles and click the Remove button.

Click the **Compute** button to examine the current list of target platform bundles and determine what other bundles need to be added to the target platform to resolve any package or service dependencies. A series of dialogs will appear that will denote which bundles should be added to the platform to fulfill dependencies and which bundles should be removed due to conflicting packages. As the dialogs are displayed, you can choose to select bundles and click **Resolve...** to apply changes, or click **Ignore...** and skip over the suggested bundles.

Summary page

The Summary page reviews all of the selections that have been made in the wizard. If the **Build platform** checkbox is selected, a target platform build process will be started immediately after the Platform Builder project is created.

Customizing a Platform Builder project

The Platform Builder Properties Editor is a multi-page editor that launches when you double-click the `eswe.properties` file in the created project and you select **Platform Builder Properties Editor** from the popup menu.

Platform panel

The controls on this panel are the same as in the Platform Options page in the Platform Builder project wizard.

J9 panel

The controls on this panel are the same as in the J9 Options page in the Platform Builder project wizard. If the JVM type on the Platform Options panel is J9, the editor enables the controls on the J9 Options page. If the JVM type is Custom, all of the options on this page will be disabled.

Startup panel

The controls on this panel are the same as in the Startup Options page in the Platform Builder project wizard.

Languages panel

The controls on this panel are the same as in the Languages page in the Platform Builder project wizard.

Custom Bundles panel

The controls on this panel are the same as in the Custom Bundles page in the Platform Builder project wizard.

System Bundles panel

The controls on this panel are the same as in the System Bundles page in the Platform Builder project wizard.

Building a target platform

To build a target platform image from a Platform Builder project, perform the following steps:

1. Right click the project and select **Extension Services > Build Platform**.
2. If you want to change the build output location, please provide a new location in the **Output location** field.
3. Click **Finish**.

After the build completes, the build file named `platform.xxx` will be placed into the output location you specified (where xxx represents the output format extension you chose, such as zip). The Platform Builder will also include the `platform.xxx` file in the project's `output` directory and a project `temp` directory will be created that contains the pre-packaged files from the build.

Chapter 2. Samples

Calendar

The Calendar sample is a simple calendar/scheduling application that is accessed by a web browser. It uses servlet technology, JDBC libraries and a local DB2e database. This sample shows how to take an existing application project and run it within WebSphere Studio. Additional steps demonstrate how to use the Extension Services Platform Builder to generate and run a target platform.

Creating the project

Perform the following steps to create the Calendar project:

1. Select **File > New > Example... > Extension Services > Calendar**. Click **Next**.
2. Click **Finish**.

Note: If prompted to switch to the SMF perspective, click **OK**.

Running in WebSphere Studio

Setting up

The Calendar sample requires the use of native libraries which WebSphere Studio must be able to locate. If you have not already completed the steps in the Getting Started section of this guide, perform the following steps to set up the environment for the Calendar sample:

1. On Windows, add the following entry to the end of your `PATH` environment variable:

   ```
   WSDD_HOME\wsdd5.0\technologies\eswe\files\db2e\win32\x86
   ```

   On Linux, add the following entry to the end of your `LD_LIBRARY_PATH` environment variable:

   ```
   WSDD_HOME/wsdd5.0/technologies/eswe/files/db2e/linux/x86
   ```

   Where `WSDD_HOME` is the location where you installed WebSphere Studio Device Developer 5.7.
2. Restart WebSphere Studio.

Deploying the application

Perform the following steps to submit the Calendar application bundle to an SMF bundle server:

1. In the SMF perspective, Select **Run > Run...** to display the Launch Configurations window.
2. Select **SMF Bundle Server**. Click **New**. Provide a name for the bundle server and keep all other default settings. Click **Run** to start the bundle server. For more information on the available bundle server options, refer to the Service Management Framework Bundle Developer Tools User's Guide.
3. Right click on the **Calendar** project and select **SMF > Submit Bundle...**
4. In the resulting Target Submission window, select **Submit Jar**.
5. Select **Admin@localhost:8080/smf** in the Export Targets box.
6. Click **Finish** to submit the bundle to the bundle server.
7. Switch to the Bundle Server view and open **Admin@localhost:8080/smf > Bundles**. Verify that a bundle named `CalendarSample` is listed.

Launching the application

Perform the following steps to install and run the Calendar sample application on an SMF runtime:

1. In the SMF perspective, select **Run > Run ...** to display the Launch Configurations window.
2. Select **SMF Runtime**. Click **New**. Provide a name for the runtime and keep all other default settings. Click **Run** to start the runtime. In the SMF Runtime view, an active runtime is shown that contains only an SMF System Bundle. For more information on the available runtime options, refer to the Service Management Framework Bundle Developer Tools User's Guide.
3. Switch to the Bundle Server view and open **Admin@localhost:8080/smf > Bundles**. Right click the **CalendarSample** bundle and select **Install**. The CalendarSample bundle, along with all of its dependent bundles, will be installed onto the SMF runtime.

For instructions on how to access and use the Calendar sample application, see "Using the application" on page 9.

Running on a device

Deploying the application

Perform the following steps to submit the Calendar sample application bundle to the file system:

1. In the SMF perspective, right click on the **Calendar** project and select **SMF > Submit Bundle...**
2. In the resulting Target Submission window, select **Submit Jar** and click **Add Directory...**
3. Select a folder in which to submit the Calendar sample application bundle.
4. Select the newly added directory in the Export Targets box.
5. Click **Finish** to submit the bundle to the filesystem.
6. Verify that the bundle are placed into the specified directory. The bundle will be named `CalendarSample+5_7_0.jar`.

Creating the Platform Builder project

Perform the following steps to create a Platform Builder project that contains the Calendar sample application bundle. This project can be used to build a target platform for a specific device type that runs the Calendar sample application.

1. Select **File > New > Project > Extension Services > Platform Builder**. Click **Next**.
2. Enter `CalendarPlatformBuilder` for the project name. Click **Next**.
3. On the Platform Options page, select the desired device type. Select the desired output format and specify an output location. If the output location is left blank, the build output will be placed in the `CalendarPlatformBuilder/output/` directory. Click **Next**.
4. On the J9 Options page, select the **jclFoundation** class library and the desired package format.
5. On the Startup Options page, leave all options with their default values. Click **Next**.
6. On the Platform Languages page, select the desired platform language support. Click **Next**.

7. On the Custom Bundles page, click **Add...** and select the **CalendarSample+5_7_0.jar** file from the filesystem location where it was deployed. Click **OK**.
8. Click the **Compute** button to determine the dependencies for the Calendar sample application bundle. A Resolve Dependencies dialog will appear. Select the following bundles from the list and then click **Resolve...**
   - DB2 Everyplace Client
   - JDBC 2.0
   - Java Servlet 2.1 API
   - OSGi HTTP Service
9. After recalculating the platform dependencies, another Resolve Dependencies dialog will appear. Click **Select All** and then click **Resolve...** A dialog will appear stating that all dependencies have been resolved. Click **OK**. Click **Next**.
10. On the System Bundles page, the system bundles that were added from the dependency resolution are listed. Click **Next**.
11. On the Summary page, make sure that the **Build platform** checkbox is selected. Review the summary and click **Finish**. The wizard will create the `CalendarPlatformBuilder` project in the workspace and will immediately launch a build to generate a target platform.
12. When the build is complete, verify that the target platform package is placed in the previously specified output directory.

Launching the application

To run the target platform on your device, perform the following steps:

1. Install the target platform onto your device. This procedure may differ by device, depending on which output format was selected:
   - If the output format was zip, tar or tar.gz, unpackage the platform package into a location on the device filesystem
   - If the output format was ipk, copy the ipk file to the device and install with the appropriate iPKG tools
2. Execute the platform startup script. The startup script is named `StartSMF` and is located in the `smf` directory of the target platform. Running this script will launch the Extension Services platform.

   Note: If running on a Linux device (Red Hat PC or Zaurus), be sure to run the StartSMF script as the root user, as the platform will be attempting to access port 80. See the Additional Information appendix in this guide for more information.

For instructions on how to access and use the Calendar sample application, see "Using the application."

Using the application

Perform the following steps to access and use the Calendar sample application:

1. Open a browser and go to http://localhost/Calendar to access the Calendar sample application.
2. The current date is displayed, along with a range of times. At the bottom of the page is a row of icons that allow for the switching of views (daily, weekly, monthly, today), creation of new events, calendar settings and help.
3. Clicking on a time (or clicking the **New Event** icon) displays the New Event page. This page allows for the creation of a new event in the calendar. A

summary and event type can be entered, along with the date of the event and the duration. Click **Add Event** to add this event to the calendar or Cancel to return to the previous view.

4. The Calendar Settings page allows for the customization of the default time grid and default view, as well as the colors for the pages that are displayed.