



# **Integrated Development Environment Build Process**

Writer: Travis Hobrla

Published: March 2011

Applies To: Windows Embedded Compact 7

## **Abstract**

Describes how to use the Windows Embedded Compact 7 integrated development environment (IDE) to configure and build your Windows Embedded Compact 7 OS designs.

- Build process overview
- Building with Platform Builder
- IDE elements for configuring and optimizing the build process
- IDE build process customization

# Contents

Introduction .....	3
Build Process Overview.....	3
Common Terminology.....	4
Building with Platform Builder .....	5
Deeper into the IDE.....	6
Making Selections in the Catalog .....	6
Adding and Configuring Subprojects .....	7
Creating SDKs.....	7
Build Commands.....	8
Build Solution.....	8
Rebuild Solution.....	8
Build Current BSP and Subprojects.....	8
IDE Build Process Customization .....	8
Targeted Builds.....	8
Release Directory Modules .....	9
Alternate Release Directories.....	9
OS Design and Build Options.....	10
OS Design Configuration Properties.....	10
Configuration Properties (General).....	11
Configuration Properties (Locale).....	11
Configuration Properties (Build Options) .....	11
Configuration Properties (Environment).....	11
Configuration Properties (Custom Build Actions) .....	11
Configuration Properties (Subproject Image Settings) .....	11
Batch Building and Scripting.....	11
Conclusion .....	12
Additional Resources .....	12

# Introduction

---

You can create Windows Embedded Compact 7 OS designs using a rich graphical user interface. Platform Builder and Visual Studio 2008 make up an integrated development environment (IDE) where you can build, develop, download, and debug code for an embedded device.

To get you started, this article provides an overview of the build process and some common terms used in the IDE. You will learn to create an OS design from a design template, customize its components, and then build it.

By using Visual Studio Solution Explorer, you can add and manage OS design subprojects and SDKs. You can also add and remove both catalog items and BSPs from your OS design as needed.

When you build using the IDE, Platform Builder links and copies all necessary OS components, builds the platform, builds subprojects, copies files to the release directory, and makes the run-time image.

Platform Builder gives you several options for building your OS design. Your choices range from running the complete build process to building a single directory or subproject. Each build option is described, and guidance is given for how to choose the appropriate build command for your needs.

You can customize your build process with time-saving functionality such as targeted building, Release Directory Modules, Alternate Release Directories, batch building, and scripting. By using the various OS design configuration property pages, you gain control over configuration settings that affect your build.

The scope of this technical article is limited to building OS designs with the Platform Builder IDE. See the [Additional Resources](#) section for links to resources on debugging your OS design, connecting a completed build to a device, building from the command line, and differences between building in Windows Embedded Compact 7 and building in previous versions of Windows Embedded CE.

## Build Process Overview

---

Before you use Platform Builder in the Visual Studio IDE, it is helpful to have an understanding of the build process and some commonly used terms.

The Windows Embedded Compact 7 build system is responsible for building an entire operating system. Unlike building a software application, building an entire operating system includes a large variety of components:

- The core operating system (kernel, file system, security features)
- Built-in applications and services (Media Player, Internet Explorer, Remote Desktop Connection)
- Built-in driver stacks (USB, networking)
- Custom drivers and hardware abstractions (display driver, audio driver, OEM adaptation layer)
- Custom applications

The first objective of the build system is to combine all of these components into a single binary image. An image is typically a single file that contains all of the above components laid out in memory so that they can be downloaded to a device and then executed by the device. The second objective of the build system is to allow components to be easily selected and substituted so that only the necessary components are included in the image.

You can use a Windows Embedded Compact 7 OS on a wide variety of embedded devices. Some devices may not take advantage of a particular operating system component. For example, a device that has no display or audio hardware has no use for a media player application. If the build system included a media player application regardless, it would waste memory on the device that otherwise could be used to increase performance or reduce cost. The build system must make it easy to build only the necessary components.

## Common Terminology

The following is a summary of some common terms you will encounter as you use the Platform Builder IDE.

### OS Design

The equivalent of a Visual Studio project. It is a collection of files and build parameters that inform the build process. To build using the Platform Builder user interface (UI), you must create an OS design or open an existing one.

### Catalog

A graphical representation of all of the available components in the Windows Embedded Compact 7 OS. Typically, these components are not hardware-specific. Each OS design has a set of catalog items associated with it. By selecting catalog items, you can include or exclude OS functionality. When you exclude components, they are not built by the build process or included in the final image.

### Board Support Package (BSP)

The hardware-specific code that adapts the OS to a specific device. You can associate one or more BSPs with an OS design.

### Subproject

A project that exists beneath the OS design in the Solution Explorer hierarchy. A subproject depends on components or files that are generated when the OS design is built. You typically use subprojects to implement applications that will be deployed to the device.

### Software Development Kit (SDK)

A set of libraries that a developer can link to even without access to the OS design. An SDK is not built during the normal build process, but rather through a separate process discussed in the [Creating SDKs](#) section. The SDK represents the OS components, BSPs, and subproject functionality that is present in the OS design.

# Building with Platform Builder

---

With an understanding of the build terminology in the Platform Builder interface, you can examine these concepts in the context of the build process by walking through a typical build. The first step is to open an existing OS design or create a new one.

## To create a new OS design

1. In Visual Studio 2008, on the **File** menu, click **New**, and then click **Project**.
2. In the **New Project** dialog box, select **Platform Builder** as the project type and **OSDesign** as the template. Click **OK**.
3. In the Create an OS Design Wizard, on the **Welcome to the OS Design Wizard** page, click **Next**.
4. On the **Board Support Packages** page, select one or more BSPs for your OS design, and then click **Next**.
5. On the **Design Templates** page, select a basic template for the device, and then click **Next**.
6. Based on the design template that you chose, the wizard presents different OS component options that will work with that template. Select the options that you want as you step through the wizard.
7. On the final page of the wizard, click **Finish**.
8. Depending on the catalog items that you included in your OS design, the **Catalog Item Notification** dialog box may appear to explain security warnings associated with catalog items. After you have read and understand these warnings, click **Acknowledge**.

Your initial catalog item selections are simply a starting point for your OS design. You can add and remove catalog items throughout development. For more information, see the section [Making Selections in the Catalog](#).

After you finish the Create an OS Design Wizard, you have an OS design that you can build. Although Platform Builder provides various build options, you must run a complete build the first time that you build your OS design.

## To run a complete build

- In Visual Studio 2008, on the **Build** menu, click **Build Solution**.

Below is a brief description of each step performed during a full build:

1. **Sysgen OS:** Based on the selected catalog items, links and copies all necessary OS components, outputting them to the SYSGEN output root subdirectory of the OS design, \$(SG\_OUTPUT\_ROOT).
2. **Build platform:** Based on the BSPs selected during OS design creation, the build process compiles, links, and copies all necessary hardware-specific code to \$(SG\_OUTPUT\_ROOT).

3. **Build subprojects:** For each subproject of the OS design, the build compiles, links, and copies all necessary binaries.
4. **Copy Files to Release Directory:** The build process copies all of the needed files from the prior steps to the Flat Release Directory, which is a single directory that contains all of the files to be included in the final image. This step can be run independently from the Build menu.
5. **Make Run-Time Image:** Based on the selections in the Catalog, the build process combines all of the OS components, hardware-specific components, and applications into a single image that can be downloaded to the device. This step can be run independently from the Build menu.

When Platform Builder performs these tasks, it actually runs command-line tools to perform the build steps. The Sysgen OS phase links and copies but does not compile because Platform Builder comes with precompiled binaries for all of the OS components. For more detailed information about this process, see [Windows Embedded Compact 7 Build Process](http://go.microsoft.com/fwlink/?LinkID=208071) (<http://go.microsoft.com/fwlink/?LinkID=208071>).

## Deeper into the IDE

---

With an understanding of how Platform Builder builds an image, you can examine some specific IDE elements that you can use to configure and optimize the build process.

### Making Selections in the Catalog

You can add and remove both catalog items and BSPs for your OS design in the **Catalog Items View** of Solution Explorer.

#### ▶ To add or remove a catalog item

1. In Solution Explorer, click the **Catalog Items View** tab.
2. Expand the CoreOS folder to see the hierarchy of all available catalog items.
3. Select a catalog item to add it, or clear the check box to remove the catalog item.

Selected catalog items are explicitly included. A green box next to a catalog item indicates that another catalog item depends on it; the catalog item with the green box will be included as long as its dependent item is included. You can get more information about dependencies by right-clicking a catalog item and clicking either **Reasons for Inclusion of Item** or **Show Dependencies**.

#### ▶ To add or remove a BSP

1. In Solution Explorer, click the **Catalog Items View** tab.
2. Expand the BSP folder to see the hierarchy of all available BSPs.
3. Select to add a BSP, or clear the check box to remove a BSP.

## Adding and Configuring Subprojects

### ▶ To add a new subproject

1. In Solution Explorer, right-click the Subprojects folder.
2. Click **Add New Subproject**.
3. On the **Select name, location and template** page of the Subproject Wizard, select a template from the **Available templates** list.
4. Click **Next**.
5. On the **Auto-generated subproject files** page, select the option for the level of automatically generated code that you would like.
6. Click **Finish**.

You can also add existing subprojects or change the build order of subprojects.

### ▶ To change the configuration settings of a subproject

1. In Solution Explorer, right-click an existing subproject.
2. Click **Properties**.
3. In the dialog box that appears, change property values as needed.

## Creating SDKs

You can create SDKs for your OS design that an application developer can use to target the device that your OS design supports.

### ▶ To create a new SDK

1. In Solution Explorer, right-click the SDKs folder.
2. Click **Add New SDK**.
3. In the **Property Pages** dialog box, set the values that are required. You can also set values for other, optional, SDK properties, such as **CPU Families** that are supported and **License Terms**.
4. Click **OK**.

You can also use a previously generated SDK.

### ▶ To add an existing SDK

1. In Solution Explorer, right-click the SDKs folder.
2. Click **Add Existing SDK**.
3. Browse to the folder for the SDK that you want to add.

4. Select the .sdksfg file for the SDK that you want to add.
5. Click **Open**.

## Build Commands

You can run a variety of build commands from the Platform Builder IDE. Some of the most commonly used build options found on the **Build** menu are:

### Build Solution

Runs the complete build process. This option is useful when you build your OS design for the first time.

### Rebuild Solution

Deletes the contents of \$(SG\_OUTPUT\_ROOT), the SYSGEN output root directory, before executing the build process from start to finish. This option is useful if you have already built your OS design but have changed catalog items.

### Build Current BSP and Subprojects

Found under Advanced Build Commands, this command starts the build at the build platform step, foregoing the Sysgen OS step. If you've already run Build Solution once and haven't changed catalog items, you can use this option to save time.

## IDE Build Process Customization

---

In addition to providing access to the different build commands, the Platform Builder IDE supports several ways to customize your build process.

### Targeted Builds

In many situations, it is unnecessary to run the entire build process. By using a targeted build, you can save time by building specific directories or subprojects.

#### To run a targeted build on a directory or subproject

- In Solution Explorer, right-click the item, and then click **Build**.

Note that to get these changes represented in the final image, you need to rebuild the image.



▶ **To manually rebuild the image**

- On the **Build** menu, click **Make Run-Time Image**.

You can set the image to rebuild automatically (so you no longer need to run it separately).

▶ **To enable automatic image rebuilding**

1. On the **Build** menu, click **Targeted Build Settings**.
2. Select **Make Run-Time Image After Building**.

You may want your targeted builds to include debug messages.

▶ **To enable debug binaries for targeted builds**

1. On the **Build** menu, click **Targeted Build Settings**.
2. Select **Always Build Debug**.

## Release Directory Modules

When you specify a Release Directory Module, Platform Builder loads the .exe or .dll file at runtime from the Flat Release Directory instead of using the .exe or .dll file from the image. Release Directory Modules are an alternative to running the **Make Run-Time Image** command after a targeted build. This approach is especially useful when debugging applications, because you can avoid rebuilding and downloading a new image to the embedded device.

▶ **To add a Release Directory Module**

1. On the **Target** menu, click **Release Directory Modules**.
2. In the **Release Directory Modules** dialog box, click **Add**.
3. In the **Module** box, enter the name of the file to add, or select it from the drop-down list.
4. Click **OK**, and then click **OK** again.

## Alternate Release Directories

When you specify an alternate release directory, Platform Builder checks it for files that are not found in the primary release directory. Alternate release directories are useful if you use Release Directory Modules to load files from multiple locations and you want to create a priority list of search paths.

▶ **To add an Alternate Release Directory**

1. On the **Target** menu, click **Alternate Release Directories**.

2. Click the **Add new directory** button.
3. Select the folder that you want to add.
4. Click **OK**, and then click **Apply**.

## OS Design and Build Options

There are several fundamental build options that you can change for your OS design.

### ▶ To change OS Design and Build Options

1. On the **Tools** menu, click **Options**.
2. In the **Options** dialog box, in the left pane, double-click **Platform Builder** to expand it, and then double-click **OS Design and Build** to expand it.
3. Select a page under **OS Design and Build** to modify the settings on that page.

For example, on the **General** page, you can change the OS build tree and the number of processor cores to use during the build. On the **Build Commands** page, you can alter the command lines that are executed when you run the **Build** or **Rebuild Solution** commands. On the **Verbosity** page, you can disable some build warnings.

## OS Design Configuration Properties

You can manage several OS design configuration properties that affect your build. Two methods for viewing the OS design configuration property pages are explained below.

### ▶ To view the OS design Configuration Properties using the context menu

1. In Solution Explorer, right-click the OS design project and click **Properties**.
2. In the **Property Pages** dialog box, if the **Configuration Properties** page is not expanded, double-click the page to expand it.

### ▶ To view the OS design Configuration Properties using the menu bar

1. In Solution Explorer, select the OS design project.
2. On the **Project** menu, click **Properties**.
3. In the **Property Pages** dialog box, if the **Configuration Properties** page is not expanded, double-click the page to expand it.

Each of the OS design configuration property pages are discussed below.

## Configuration Properties (General)

You can set the name of the image and location of the Flat Release Directory on the **General** page. If you change either of these properties, you will need to re-run the **Copy Files to Release Directory** command.

You can also select the build type: **Debug**, **Release**, or **Checked**. A debug image is useful for development because the image includes additional debug output. With a debug image you can perform full source-level debugging because compiler optimizations are turned off. For more information about the differences between Debug, Release, or Checked builds, see [Windows Embedded Compact 7 Build Process](http://go.microsoft.com/fwlink/?LinkID=208071) (<http://go.microsoft.com/fwlink/?LinkID=208071>).

## Configuration Properties (Locale)

You can use the **Locale** page to set environment variables that are passed to the build localization tools during the build process. In this way, you can create an OS design for non-English languages.

## Configuration Properties (Build Options)

You can set several environment variables on the **Build Options** page that affect how the image behaves, such as enabling the kernel debugger and profiling.

## Configuration Properties (Environment)

You can set custom environment variables to influence the build on the **Environment** page. You typically use these variables in conjunction with custom build logic in a hardware platform or subproject. By default, none are set because this functionality is typically for advanced users.

## Configuration Properties (Custom Build Actions)

You can set actions (typically batch files) to be called during the various build steps by using the **Custom Build Actions** page. Custom Build Actions are typically for advanced users, and, by default, no actions are set.

## Configuration Properties (Subproject Image Settings)

You can use the **Subproject Image Settings** page to include or exclude a subproject from the build or the final image. This page is not the same as the Subproject Properties page that you access by right-clicking a subproject in Solution Explorer.

## Batch Building and Scripting

You can automate the building of multiple configurations of an OS design sequentially by using the **Batch Build** command.

▶ **To set up and run a batch build**

1. On the **Build** menu, click **Batch Build**.
2. In the **Batch Build** dialog box, in the **Build** column, select each build configuration that you want to add to your batch build list.
3. Click **Build**.

By using the **Build** and **Rebuild** buttons in the **Batch Build** dialog box, you can run the **Build Solution** and **Rebuild Solution** commands for multiple configurations in series. For more granular control, you can generate scripts that run many IDE commands automatically. You can also make command-line calls from scripts.

▶ **To access the script editor**

1. On the **Tools** menu, click **Platform Builder**.
2. Click **Configure Scripts**.

For information about using the script editor, see the [Configure Scripts Dialog Box](http://go.microsoft.com/fwlink/?LinkId=209871) (<http://go.microsoft.com/fwlink/?LinkId=209871>) reference topic.

## Conclusion

---

The IDE composed of Platform Builder and Visual Studio 2008 is a unified environment where you can build, develop, download, and debug your Windows Embedded Compact 7 OS designs. You can manage many aspects of your OS design from Solution Explorer, including catalog items, BSPs, subprojects, and SDKs. A build can vary in scope from building the entire OS design to building a single directory or subproject. You can customize your build process with time-saving functionality such as targeted building, Release Directory Modules, Alternate Release Directories, batch building and scripting. By using the various OS design configuration property pages, you gain control over configuration settings that affect your build. The Platform Builder IDE gives you a full suite of tools you can use to create Windows Embedded Compact 7 OS designs within a convenient graphical user interface.

## Additional Resources

---

- [Debugging and Diagnostics](http://go.microsoft.com/fwlink/?LinkId=208572) (<http://go.microsoft.com/fwlink/?LinkId=208572>)
- [Getting Started with Virtual CEPC, see “Prepare Platform Builder for Boot Image Download”](http://go.microsoft.com/fwlink/?LinkId=205781) (<http://go.microsoft.com/fwlink/?LinkId=205781>)
- [Windows Embedded Compact 7 Build Process](http://go.microsoft.com/fwlink/?LinkId=208071) (<http://go.microsoft.com/fwlink/?LinkId=208071>)
- [What's New in Platform Builder](http://go.microsoft.com/fwlink/?LinkId=205799) (<http://go.microsoft.com/fwlink/?LinkId=205799>)

- [Configure Scripts Dialog Box](http://go.microsoft.com/fwlink/?LinkId=209871) (http://go.microsoft.com/fwlink/?LinkId=209871)
- [Windows Embedded website](http://go.microsoft.com/fwlink/?LinkId=183524) (http://go.microsoft.com/fwlink/?LinkId=183524)
- [Windows CE Base Team Blog](http://go.microsoft.com/fwlink/?LinkId=205449) (http://go.microsoft.com/fwlink/?LinkId=205449)

This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2011 Microsoft. All rights reserved.