

LINEO™  
**Embedix™**  
SDK



*Target Wizard  
User Guide*



LINEO™





L I N E O™

*Embedix SDK 2.4  
Target Wizard  
User Guide*

## **Disclaimer**

Lineo, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Lineo, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Lineo, Inc. makes no representations or warranties with respect to any Lineo software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Lineo, Inc. reserves the right to make changes to any and all parts of Lineo software, at any time, without any obligation to notify any person or entity of such changes.

## **Trademarks**

Lineo and Embedix are registered trademarks of Lineo, Inc. The stylized Lineo logo is a trademark of Lineo, Inc.

Other product and company names mentioned in this document may be the trademarks or registered trademarks of their respective owners.

## **Copyright Information**

Copyright © 2002 Lineo, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Lineo, Inc.  
588 West 400 South  
Suite 150  
Lindon, UT 84042  
USA  
<http://www.lineo.com>

Embedix SDK 2.4 User Guide  
Part Number: EMBD-SDK-TW-0502  
May 2002

# Contents

---

<b>P R E F A C E</b>	<b>About This Guide</b> .....	v
	Conventions Used in This Document.....	v
	Admonitions.....	v
	Key Combinations.....	vi
	Special Fonts and Capitalization.....	vi
	Additional Resources.....	vii
<b>CHAPTER 1</b>	<b>Getting Started with Target Wizard</b> .....	1
	Installing Target Wizard .....	1
	Development Process Overview .....	2
	Starting and Stopping Target Wizard .....	3
	Starting Target Wizard on a Linux Machine.....	3
	Stopping Target Wizard on a Linux Machine .....	4
	Starting Target Wizard on a Windows Machine.....	5
	Stopping Target Wizard on a Windows Machine.....	6
<b>CHAPTER 2</b>	<b>Working with Projects</b> .....	9
	Creating a Project .....	9
	Saving a Project .....	12
	Opening a Project .....	12
	Closing a Project.....	13
	Backing Up a Project.....	13
	Removing a Project.....	13
	From a Linux Host.....	14
	From a Windows Host.....	14
<b>CHAPTER 3</b>	<b>Working with State Files</b> .....	15
	Sample Use of State Files .....	15
	Creating a State File .....	16
	Loading a State File .....	17

	State Files Available for “Generic x86” Targets.....	17
<b>CHAPTER 4</b>	<b>Exploring Target Wizard.....</b>	<b>19</b>
	Menu Bar, Tool Bar, and Status Bar .....	19
	The Menu Bar .....	19
	The Tool Bar .....	22
	The Status Bar .....	23
	Node List Window.....	23
	Node List Columns.....	24
	Tabs (Information Areas) .....	24
	Node Status Tab.....	25
	Description Tab .....	26
	Files Tab .....	26
	Size Tab.....	26
	Conflicts Tab.....	26
	Messages Tab.....	26
	Build Log Tab.....	26
	Navigation Tips .....	27
	General Options .....	27
	Context-Sensitive Interactions.....	27
	Target Wizard Default Settings.....	28
<b>CHAPTER 5</b>	<b>Customizing Projects .....</b>	<b>29</b>
	Modifying the Current Project .....	29
	Resolving Package Dependencies and Conflicts.....	30
	Managing Project Size .....	31
	Reducing Libraries on Your Target.....	31
	Including Symbols Explicitly .....	33
	Merging Your Application into the Target Image .....	34
	Converting Your Application into a Custom Embedix Package ..	35
<b>CHAPTER 6</b>	<b>Building and Deploying a Target Image.....</b>	<b>37</b>
	Building a Target Project.....	37
	Changing Project Build Options .....	39
	Deploying a Target Image—General Guidelines .....	39
	Deploying a Target Image to an x86 Target .....	40
	Start the x86 Deployment Wizard .....	40

	x86 Deployment Options.....	42
	Make a Directory Tree .....	44
	Make a Self-Hosting RAM Disk Filesystem.....	44
	Make a Filesystem to chroot Into.....	45
	Make Install Disks for a RAM Disk System.....	45
	Make Install Disks for a Normal Filesystem.....	47
	Make a Bootable Install CD .....	48
	Make a CD that Boots to the OS .....	49
	Deploying to and from a VM-ware Hosted Virtual Linux Machine (SDK for Windows Users Only).....	50
	Deploying to a External Hardware Target .....	51
	Deploying using Floppy Disks .....	51
	Sharing Files with the Windows File System .....	52
<b>A P P E N D I X A</b>	<b>Exploring a Sample Project.....</b>	<b>55</b>
	About the Host and Target .....	55
	Creating the Sample Project .....	56
	Choosing and Customizing a Configuration Profile .....	62
	Exploring the Tree .....	65
	Resolving Dependencies .....	66
	Introducing and Resolving Conflicts into the Tree.....	70
	Checking Project Status .....	73
<b>A P P E N D I X B</b>	<b>Files and Directories .....</b>	<b>75</b>
	Project Files and Directories.....	75
	Packages .....	76
	build .....	77
	config-data .....	77
	merge .....	77
	Target Wizard Defaults Files .....	78
	Linux System Subdirectories .....	80
	/opt/Embedix/embedix-2.0/Packages .....	80
	/opt/Embedix/embedix-2.0/config-data/ecds .....	80
	/opt/Embedix/embedix-2.0/config-data/build-control.....	80
	/opt/Embedix/embedix-2.0/config-data/patches.....	81
	/opt/Embedix/embedix-2.0/config-data/preconfigs.....	81
	/opt/Embedix/tools/ .....	81
	/opt/Embedix/emb-bin.....	81

	Package Groups in Target Wizard .....	82
<b>APPENDIX C</b>	<b>Windows Build Server Information .....</b>	<b>87</b>
	Finding a Build Server.....	87
	Adding a Build Server Manually .....	87



## About This Guide

---

This preface includes information on formatting practices used throughout this Lineo<sup>®</sup> Embedix<sup>®</sup> document.

### Conventions Used in This Document

The style conventions used in the printed and PDF format of this document do not necessarily apply to other formats. During conversion to HTML, some of these conventions may be lost.

This document uses the following graphical and typographical conventions:

- ▶ Admonitions
- ▶ Key combinations
- ▶ Special fonts and capitalization

#### Admonitions

Note, Tip, and Warning paragraphs draw your attention to additional information which may help you avoid losing data or time.



**Note:** Notes contain additional information about the current topic.

---



**Tip:** Tips contain suggestions that may save you time or effort.

---



**Warning:** Warnings contain critical information that you need to understand before proceeding. Ignoring information in a warning may cause loss of data or time.

---

## Key Combinations

Key combinations (such as Ctrl+O) are presented throughout this document and should be used in the following way:

1. Press and hold the first key.
2. Press the second key.
3. Release both keys.

## Special Fonts and Capitalization

Two special fonts have been used to distinguish between user input and computer output:

### ▶ **Command**

All commands or data to be entered on an on-screen data entry line appear in bolded **Courier** font. This may include commands used with options, paths to directories or files, or other simple input, such as filenames.

### ▶ **Code**

Any code sample, including command output, is shown in Courier font.

The following capitalization rules apply to Linux filenames, Linux commands, and English names of on-screen buttons and keyboard keys.

- ▶ Linux filenames and commands are case-sensitive. In most instances, they are lowercase. When you enter a filename or command, use the same case that appears in your instructions or examples.

- ▶ When procedures refer to a particular on-screen button, the name of the button appears in uppercase (such as in SAVE), regardless of how it appears on the screen.
- ▶ When procedures refer to a particular key on a keyboard, only the initial key is capitalized (such as the Tab key), just as it appears on a U.S. standard keyboard. This also applies to key combinations.

## Additional Resources



---

**Note:** Most printed manuals that ship with Lineo products are also available in PDF and HTML formats on the product CD-ROM unless otherwise stated.

---

The following resources are available to provide you with additional support.

- ▶ ***Embedix SDK Getting Started*** book:  
Available in print and PDF only in this release.
- ▶ ***Embedix SDK Tools*** book:  
Available in print and PDF only in this release.
- ▶ ***Embedix SDK Reference Manual***
- ▶ ***Embedix RealTime Programming Guide***
- ▶ ***Embedix Board Support Package <Boardname> User Guide.***  
For PDF and HTML versions, see the <Boardname> CD-ROM.
- ▶ ***Embedix SDK DDD Manual:***  
Available on the SDK CD-ROM only in this release.
- ▶ **Lineo Support Web site:**  
<http://www.lineo.com/services/support/index.html>
- ▶ **Lineo Developers Support Web Pages:**  
<http://members.lineo.com/dev-wiki>

## Additional Resources

---

# Getting Started with Target Wizard

---

Lineo® Embedix® Target Wizard assists you in developing and deploying highly customized embedded systems that run Embedix. Target Wizard helps you to manage package dependencies, remove conflicts between components, integrate applications with the operating system, and reduce the size of your embedded target image.

Target Wizard is included with both Linux and Windows versions of the Embedix SDK. Where required, Linux-specific instruction and Windows-specific instruction is provided throughout this manual.

This chapter introduces Embedix Target Wizard. It includes the following topics:

- ▶ “Installing Target Wizard” on page 1
- ▶ “Development Process Overview” on page 2
- ▶ “Starting and Stopping Target Wizard” on page 3

## Installing Target Wizard

Target Wizard is automatically installed when you install the SDK. For platform-specific SDK installation instruction:

**Embedix SDK** users should refer to  
*Embedix SDK Getting Started*

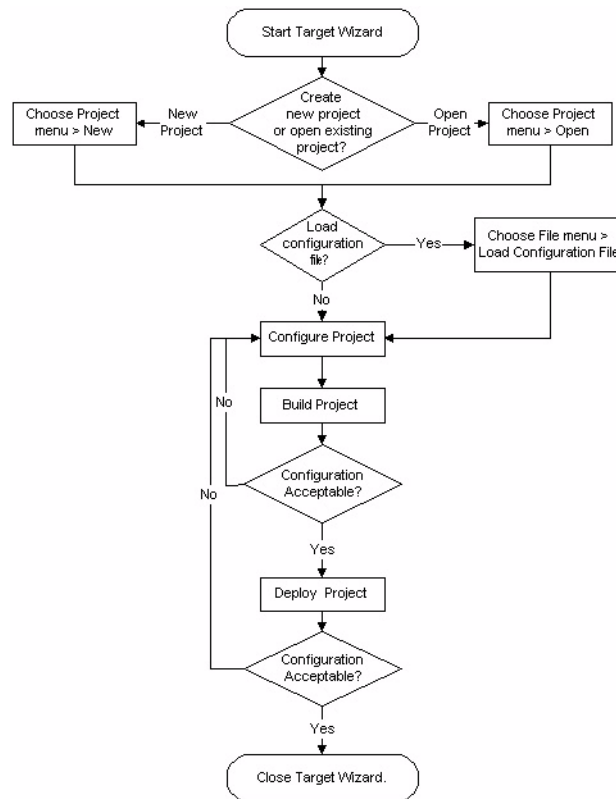
**Embedix SDK for Windows** users should refer to  
*Embedix SDK for Windows User Guide*

## Development Process Overview

Developing an embedded system involves an iterative process of customizing the Embedix operating system for the target device, creating the target's custom application(s), building and deploying the Embedix image, and evaluating the embedded system.

The flowchart shown in Figure 1-1 provides a visual overview of a typical development process.

**Figure 1-1.** Target Wizard Development Process



Target Wizard is the primary tool in the Embedix SDK for customizing, building, and deploying the Embedix operating system. Although the SDK includes a fully featured IDE, while you are customizing the Embedix project, you can use your preferred development tools to develop and debug the application code.

## Starting and Stopping Target Wizard

The procedures for starting and stopping Target Wizard depend on your host operating system—Linux or Windows. Refer to the instructions applicable to your system.

- ▶ “Starting Target Wizard on a Linux Machine” on page 3
- ▶ “Stopping Target Wizard on a Linux Machine” on page 4
- ▶ “Starting Target Wizard on a Windows Machine” on page 5
- ▶ “Stopping Target Wizard on a Windows Machine” on page 6

### Starting Target Wizard on a Linux Machine

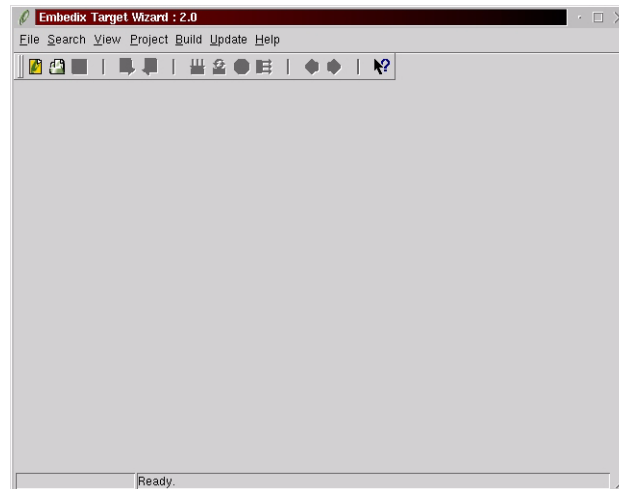
If your computer is running Linux, use the following procedure to start Target Wizard:

1. Click the Terminal Emulation (X-term) icon on your screen to access a shell prompt.
2. Start Embedix Target Wizard by entering the command: **tw**

Target Wizard displays its splash screen. Following the splash screen, Target Wizard opens on the desktop.

The first time you run Target Wizard, it displays the user interface without a project, as shown in Figure 1-2 on page 4.

**Figure 1-2.** Embedix Target Wizard User Interface (no project loaded)



Each time you start Target Wizard after you have created one or more projects, Target Wizard opens the project that was most recently opened. (See “Target Wizard Default Settings” on page 28 for more information about this option.)

For instructions on creating a Target Wizard project, see “Creating a Project” on page 9.

## Stopping Target Wizard on a Linux Machine

Use the following procedure to stop Target Wizard on Linux.

1. Save the current project.
2. Ensure that Target Wizard is not currently building or deploying a project.
3. Click the 'X' in the upper right corner of the Target Wizard application.

TW exits to the Linux desktop.

For instructions on creating a Target Wizard project, see “Creating a Project” on page 9.



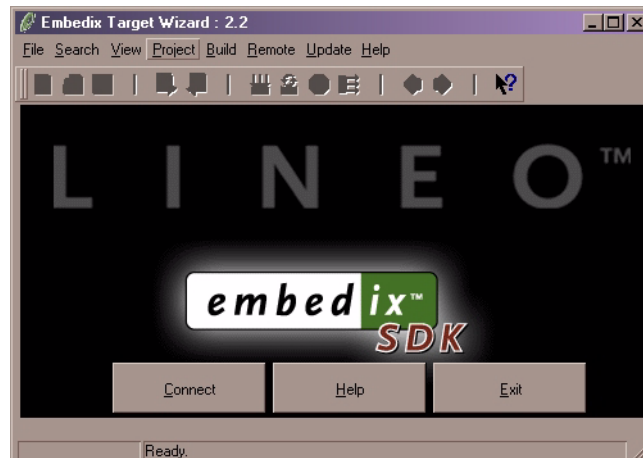
## Starting Target Wizard on a Windows Machine

If your computer is running Windows NT or Windows 2000, use the following procedure to start Target Wizard and connect to the build server:

1. Start your VMware workstation (as described in the *Embedix SDK for Windows User Guide*).
2. From the Quick Launch bar at the bottom of your screen, click Embedix Target Wizard (or, from the Start button, navigate to Start > Programs > Embedix SDK > Embedix SDK and then click the Embedix SDK executable).

The Target Wizard splash screen appears briefly, followed by the screen shown in Figure 1-3.

**Figure 1-3.** Embedix SDK for Windows, Opening Screen

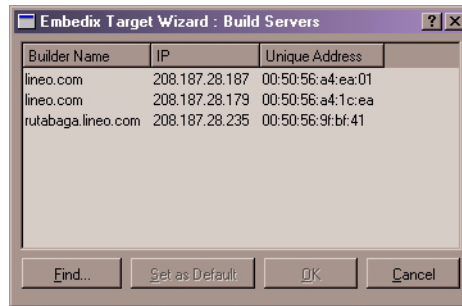


3. Click Connect to open Target Wizard's Build Servers screen.
4. On the Embedix Target Wizard Build Servers screen, do one of the following:
  - ▷ Click FIND to search the network for available Target Wizard Build Servers. Target Wizard displays a screen similar to Figure 1-4.

- ▷ Click ADD to add a build server to your network and then complete the following fields.

Builder name  
Builder IP address  
Builder port (provided)

**Figure 1-4.** Sample List of Available Build Servers



5. Highlight the appropriate build server in the list and then click OK.

Target Wizard opens on the Windows desktop and launches the Linux virtual machine.

- ▷ For instructions on creating a Target Wizard project, see “Creating a Project” on page 9.



---

**Note:** See *Embedix SDK for Windows User Guide* for an overview of the product architecture and instructions for installing, configuring, and using a build server.

---

## Stopping Target Wizard on a Windows Machine

Use the following procedure to stop Target Wizard on Windows.

1. Save your current project(s).
2. Ensure that Target Wizard is not currently building or deploying a project.
3. Press Ctrl+Q.

Target Wizard exits to the host operating system.



---

**Tip:** To save time shutting down and restarting the build server's virtual machine, click Suspend on the VMware toolbar.

---



## CHAPTER 2 Working with Projects

---

Target Wizard uses projects to organize your work. Each project is stored in its own directory, which contains a number of files and sub-directories.

This chapter contains the following sections:

- ▶ Creating a Project
- ▶ Saving a Project
- ▶ Opening a Project
- ▶ Closing a Project
- ▶ Backing Up a Project
- ▶ Removing a Project

### Creating a Project

To create a project, first ensure that Target Wizard is running and that no project is open. Then use the following procedure.

1. From the Embedix Target Wizard interface, choose Project > New (or press Ctrl+N or click the New File icon ).

Target Wizard launches the New Project Setup Wizard.

2. Complete the New Project Setup Wizard.

Use the information on the following pages as a guide. For more information, click the wizard's HELP button.

#### **Project Name and Location:**

- ▶ Project Base Directory is the subdirectory where the project you create will reside.

- ▶ Project Name is any name you want to give this new project.

The project name becomes a directory name, so only the following characters are valid characters for your project name: [A-Z], [a-z], [0-9], [·], [–], [–]

**Build Options:** Select the options you want for this project. By default, all are enabled.

- ▶ Build with conflicts allows builds to continue when conflicts exist.
- ▶ Continue building when errors occur allows builds to continue when errors occur.
- ▶ **Build Log File Options** allow you to select which type of log file you prefer:

**No Log File:** No log file produced, but you can still view the last 200 lines of build history from your project's Build Log tab.

**Static Log File:** The log file specified in the entry box will be appended to with each build unless you select the "Confirm Log Overwrite" option. If you select this option, a dialog box will open with each build to allow you to select one of the following log file options: Append, Overwrite, or Rename. The log file will reside in the current project directory unless you enter a full path to another directory.

**Multiple Log Files:** A new log file is generated in the current project directory for each build with a date and time stamp in the filename. The log file name format is: *projectname\_year\_month\_day\_hour.minute.second.logfile*

For example, if you built x86Test with this option selected just before 9:30 in the morning, the system would generate the log file:

```
x86Test_2001_5_11_9.29.59.logfile
```

**Target Options:** Choose your current project's target platform from the drop-down list of board support package (BSP) options.

The options provided in the drop-down list will vary depending on which BSPs you have installed.

3. Click FINISH to create the project. The system takes a few minutes to build the project filesystem. Once the project is created, you are returned to the Embedix Target Wizard interface. You now have a project that is populated by default with all of the available packages in this product. These packages display in package groups, but with some package groups enabled and some not.



**Note:** The project directory is created at the location you specify, but the actual project files are placed in a directory under /opt/Embedix, and the named project directory is replaced with a symbolic link (symlink).

---

All new projects are loaded with a default configuration profile (or state) for the target platform you chose in the “New Project” wizard. You can change the settings at any time by customizing the default configuration or by loading a new configuration profile.

- ▶ To load a configuration profile into your project, see “Loading a Configuration Profile into Your Project” on page 35.
- ▶ To customize this project manually, see “Customizing Your Project” on page 36.

## Saving a Project

To save the current project, use Target Wizard's Ctrl+S keyboard combination or choose Project > Save.



---

**Tip:** You can set Target Wizard to automatically save the current project whenever you close a project or exit Target Wizard. For more information, see “Target Wizard Default Settings” on page 28.

---

## Opening a Project

To open an existing project, use the following procedure:

1. From the menu bar, choose Project > Open.  
Target Wizard displays the Open dialog.
2. Use the Open dialog to browse the file system and locate the project file you want to open.



---

**Note:** Target Wizard project files use the suffix .epj.

---

3. Select the project file you want to open and then click Open.  
Target Wizard opens the project for customization and editing.

### Where to Go from Here

- ▶ To load a configuration state into your project, see “Loading a State File” on page 17.
- ▶ To customize this project manually, see “Customizing Projects” on page 29.



## Closing a Project

Only one project may be open in Target Wizard at any time. To close the current project from the menu bar, choose Project > Close (or press Ctrl+W).



---

**Tip:** You can set Target Wizard to automatically save the current project whenever you close a project or exit Target Wizard. For more information, see “Target Wizard Default Settings” on page 28.

---

## Backing Up a Project

To back up your projects, use the operating system’s copy commands, file utilities, or version control software. Use the following list as a guide to the files that should be backed up.

- ▶ Packages/ (if package edits were made)
- ▶ config-date/ecds/ (if package edits were made)
- ▶ config-date/buildcontrol/ (if package edits were made)
- ▶ build/projectstate/
- ▶ build/packagestate/
- ▶ rpmdir/BUILD/
- ▶ merge/

## Removing a Project

You can conserve disk space by removing unneeded projects.



---

**Tip:** Before removing a project, be sure that the project is backed up and is not open in Target Wizard.

---

To remove a project, follow the instructions for your host development machine's operating system, below.

### From a Linux Host

To remove a project from the Linux file system, open a terminal emulation window and use this command syntax:

```
rm -rf <project> /opt/Embedix/<project>
```

where *<project>* is the project path you entered during project creation.

For example, if you specified the directory `/home/user1/project/jukebox` when you created your project, enter the following command:

```
rm -rf /home/user1/project/jukebox /opt/  
Embedix/home/user1/project/jukebox
```

### From a Windows Host

Use the Windows Explorer to explore the Samba-mounted **vkitProjects** drive. Browse the **project\** folder to locate the subfolder for the project you want to delete.

## CHAPTER 3 Working with State Files

---

Embedix Target Wizard organizes packages and options into a hierarchical tree, presented to you in the Node list View. The state of this entire tree is saved when you save your project.

Embedix Target Wizard also supports saving and loading the configuration from portions of this tree to and from state definition files, or SDFs. This can be particularly helpful for producing multiple projects with small differences.

This chapter includes the following sections:

- ▶ “Sample Use of State Files” on page 15
- ▶ “Creating a State File” on page 16
- ▶ “Loading a State File” on page 17
- ▶ “State Files Available for “Generic x86” Targets” on page 17

### Sample Use of State Files

To see how a configuration profile or state can help you, suppose that your company produces three versions of a jukebox product: *Basic*, *Intermediate*, and *Full*. The product has been proven on one particular platform, which we'll call platform A.

Your company wants to port all versions of the product to platform B, which uses a different sound chipset than platform A.

You don't want to change the kernel configuration of each of the project files of *Basic*, *Intermediate*, and *Full* individually. Knowing that the kernel configuration will be the same for all three versions on platform B, you would proceed as follows:

1. Choose one project to start from. (For the sake of this discussion, choose the *Full* version.)
  2. Make a copy of this project by loading the original *Full* project into Target Wizard.
  3. Use the *File > Save As* menu option to save it with a different name, such as *Full-b*.
  4. Develop the project until it functions properly on platform B.
  5. With project *Full-b* open, transfer the kernel configuration to the other versions of your jukebox product by creating an SDF. Use *Embedix > kernel* as the top node. (It is convenient to create the SDF in the global “preconfigs” directory so that it is accessible to other projects.)
  6. Load this SDF into copies of the other two projects, *Basic-b* and *Intermediate-b*
- All three projects produce the same kernel.

## Creating a State File

After customizing your configuration, you may want to save some or all of your current project’s settings as a state file so that you can reuse your work.

- ▶ To save the settings of your entire current project, choose *File > Save State for Entire Project*
- ▶ To save the settings of the currently selected node and its child nodes, choose *File > Save State for Current Node*.

When the Choose one type of state file (Global Preconfig, Board Specific, Project Specific, or Other), complete the dialog box, and click **SAVE**.

Your custom state file should now appear in the State Files list when you choose *File > Load State*.

## Loading a State File

Configuration profiles or state files are included in the Embedix SDK to allow you to quickly get started on your project. You can begin by applying a configuration profile, and modify individual node settings until you are ready to save your custom project.

Target Wizard supports x86 targets by default. You can expand the target devices that Target Wizard supports by installing an Embedix Board Support Package (BSP). Each BSP contains preconfiguration files for the target platform. For more information on BSPs, refer to the accompanying documentation and the Lineo Web site.

You can save any project's configuration (or a portion of a project's configuration) as a state file and have it display as another option in your File > Load State list.

### To load a configuration profile:

1. From the menu bar, choose File > Load State.

The list of configuration profiles or states applicable to your current project's target platform appears. Note that each state in the list is defined as Global, Board, Project, or Other.

2. From the State Files list, choose a configuration profile.

## State Files Available for “Generic x86” Targets

For *Generic x86 board* target platform projects, the following configuration profiles or states are available in this product:

- ▶ **Embedix\_config\_1.0:** Embedix Linux configuration for typical x86 embedded Linux requirements.
- ▶ **Embedix\_min:** Minimal Embedix Linux configuration for an x86 embedded Linux. This simply provides the setup for a system with a shell.
- ▶ **i386\_medium:** Embedix Linux configuration for 26 packages that fit on a 16 MB RAM disk.

- ▶ **i386\_many**: Embedix Linux configuration for 47 packages. This is a good option for build testing.
- ▶ **settop\_config\_pc**: Embedix Linux configuration for typical x86 embedded Linux requirements on a set-top box.

This chapter helps you explore Embedix® Target Wizard

- ▶ Menu Bar, Tool Bar, and Status Bar
- ▶ Node List Window
- ▶ Tabs (Information Areas)
- ▶ Navigation Tips
- ▶ Target Wizard Default Settings

## Menu Bar, Tool Bar, and Status Bar

The Menu bar, Tool bar, and Status bar are standard GUI application areas.

### The Menu Bar

#### Linux Host Development Machine

If you are using a Linux host development machine, the Menu bar contains the following menus: File, Search, View, Project, Build, Update, and Help. The menu items and their functions are explained in Table 4-1.

#### Windows Host Development Machine

If you are using a Windows host development machine, the Menu bar contains the following menus: File, Search, View, Project, Build, Remote, Update, and Help. The menu items and their functions are explained in Table 4-1.

**Table 4-1.** Menu Item Functions

Menu	Menu Item	Function
File	Load State	Lists configuration profiles that you can apply to your current project.
	Save State for Entire Project	Choose to save your entire project's configuration as a configuration profile or state file (an .sdf) and have it display as another option in the Load State list.
	Save State for Current Node	Choose to save the configuration of the node you currently have selected as a configuration profile or state file (an .sdf) and have it display as another option in the Load State list.
	Run Wizard	Lists available wizards to automate Target Wizard functions.
	Import Kernel .config	Lists kernel.config files that are available for import.
	Export Kernel .config	Allow you to save your project as a kernel.config file.
	Target Wizard Settings	Set general options and project creation options.
	Exit	Exit or quit Embedix Target Wizard.
Search	Search	Open dialog box and search your project for the characters you enter.
	Repeat Search	Search again using the same criteria as the previous search.
View	Info Area	Enable or disable your view of the Info Area (or tabs).
	Toolbar	Enable or disable your view of the Toolbar.
	Statusbar	Enable or disable your view of the Statusbar.
	Show Groups	Toggle between a Groups view and a Components view in the Node list (or tree) view.
Project	New	Start the New Project wizard.



**Table 4-1.** Menu Item Functions

<b>Menu</b>	<b>Menu Item</b>	<b>Function</b>
	Open	Open an existing project.
	Open Recent Project	Choose a project from the 10 most recently opened projects.
	Save	Save the current project as an .epj (project) file and an .ess (state) file.
	Close	Close the current project.
	Options	Set options for the current project.
Build	Build	Build the current project, but do not rebuild components that have not changed. (The Messages tab becomes active and displays build progress.)
	Force Rebuild	Rebuild all project components, even if nothing has changed. (The Messages tab becomes active and displays build progress.)
	Deploy	Set the target's deployment options and deploy the project to your target.
	Stop Build	Stop the current build.
Remote*	Connect to Builder	Display the Build Servers dialog (Figure 1-4 on page 6) and allow selection of Target Wizard Build Server
	Check Build Connection	Verify that Target Wizard/NT is connected
Update**	Update Now!	Access the Lineo web site for download information.
Help**	Contents	Open a table of contents for available online help topics.
	Icon Legend	Display the node status icon legend in a separate, movable window.
	Icons	View short descriptions for node status icons.
	About	View version information about Embedix Target Wizard.

\* The Remote menu appears only in the Windows version of Target Wizard.

**Table 4-1.** Menu Item Functions

Menu	Menu Item	Function
<p>** The Update and Help menu commands require an internet browser. The Update Now! command requires an internet connection. If you are running Target Wizard on a Linux workstation and you receive an error message when you try to start the menu items Update &gt; Update Now! or Help &gt; Contents, then your Internet browser is not located where Target Wizard expects to find it (the /usr/bin directory). In order to use these menu items, you must add or update the following entries in your home directory's .twconfig file:</p>		
<pre>NETSCAPE=YES BROWSER=&lt;your path to Netscape&gt;</pre>		

---

## The Tool Bar

The Tool bar icons provide the following shortcuts.

-  Create a new project
-  Open a project
-  Save a project
-  Load a state
-  Save a state for an entire project
-  Build all packages that changed since last build+build options
-  Force a rebuild of all packages (changed or not) + build options
-  Stop the build
-  Start the Deployment wizard
-  Go back
-  Go forward
-  Activate the "What's This?" button and point to an object

## The Status Bar






The Status bar appears at the bottom of the Target Wizard window and displays messages about program activity. For example, it displays the current status of an ongoing build, it. When you save a project, the status bar displays a confirmation message. If you click and hold the mouse button on a screen element, the status bar displays a short description of the element.

## Node List Window

The Node list window (also known as the tree view) displays a list of Groups, Components, and Options that are available for the current project. Select the first line (“Embedix”) in the tree view to review the current for the entire project. Select any other line to

By default, every new project is displayed with packages organized into groups, but you can switch the view to an alphabetical list of component packages by choosing the menu item View > Show Groups.

Each node in the list is in one of seven states. Each state is indicated by a corresponding icon:

-  **Disabled** means this node won't be included in the project.
-  **Disabled parent** means that although this node has a “True” value, it won't be included in the project unless its parent is enabled and fulfilled.
-  **Enabled and Fulfilled** means that this node is enabled and that all of its dependencies have been met.
-  **Enabled and Unfulfilled** means that although this node is enabled, not all of the dependencies have been met.
-  **Conflict** means that there is a conflict between enabled packages at this node. (See “Resolving Package Dependencies and Conflicts” on page 30 for a discussion of package conflicts.)



**Enabled and Fulfilled with Unfulfilled Children** means that this node is enabled and fulfilled, but that at least one of its children is unfulfilled.



**Enabled as Module** means that this node is enabled as a module that can be dynamically loaded into the kernel.

## Node List Columns

At the top of the Node list view are five Column Headers:


- ▶ **Group, Components, and Options** lists the project's nodes in a hierarchical view.
- ▶ **Value** is the current setting of the node. This setting varies, depending on the parameter type:
  - ▷ Boolean = True or False
  - ▷ String = Any valid string that you choose from a list or enter
  - ▷ Integer or Float = Any integer or fraction of an integer (can also have a range limit specified)
  - ▷ Tristate = True, False, or Module
  - ▷ DepTristate = True, False, or Module, depending on the value of another Tristate
- ▶ **Disk Size Estimate** is the approximate space this node uses on the disk.
- ▶ **Min. Ram Size Estimate** is the approximate minimum RAM size required to run the node.
- ▶ **Type** is the type of node: Boolean, String, Integer, Float, Tristate, or DepTristate.

## Tabs (Information Areas)


Below the Node list in the Target Wizard window, there is an information area with seven tabs: Node Status, Description, Files, Size, Conflicts, Messages, and Build Log.

## Node Status Tab


The Node Status tab displays four types of information that apply to the node selected in the Node list view:


 The Check symbol next to *Enabled*, *Fulfilled*, or *No Conflicts* means that the selected node has been checked and that it is enabled or fulfilled or that it is not in conflict with another enabled node.


Three green checks mean that all is well with the selected node.

 The X symbol next to *Enabled*, *Fulfilled*, or *No Conflicts* means that the selected node is not enabled, is not fulfilled, contains conflicting enabled nodes, or is in conflict with an enabled node.

Any red X means that there may be a problem in the selected node. You may decide to build your project with conflicts and accept conflicts, for example if you have enabled two different web servers for two different purposes.

 The Happy Face icon (which displays green on-screen) indicates that this package is helping to fulfill the highlighted node.

 The Sad Face icon indicates that this package is not helping to fulfill the highlighted node.

 The Dead Face icon (which displays red on-screen) indicates that a node is missing.

When a dependency is unfulfilled, you may need to set a node value or enable a node. Instructions for fulfilling the dependency appear next to the icons.

Package dependencies may be grouped in All or Any lists. In order for to fulfill these dependencies use the following approach:

- ▷ In an All list, all listed dependencies must be fulfilled.
- ▷ In an Any list, at least one of the listed dependencies must be fulfilled.

## Description Tab

The Description tab displays a short description of the selected node.

## Files Tab

The Files tab displays all files that will be in a build to support the highlighted node.

## Size Tab

The Size tab displays a dynamic summary of the size of this node in relation to the other enabled nodes in this project. For each node you select, this tab displays the node's relative percentage of the project's total size.

## Conflicts Tab

The Conflicts tab provides a work area for resolving unacceptable conflicts.

- ▶ The Name column identifies the name of the highlighted node and indicates whether it is a Group, Component, Option, or Leaf Option.
- ▶ The Provides column identifies the package that the highlighted node provides.
- ▶ The Value column identifies the selected node's current value (which is restricted by the parameter type).
- ▶ The Component column identifies where the selected node resides, which could be a group (or package) or another component.

## Messages Tab

The Messages tab automatically displays build status during builds.

## Build Log Tab

The Build Log displays the last 200 lines of build output for the most recent build.

## Navigation Tips

The Embedix Target Wizard graphical interface supports the use of the mouse, keyboard, or both. However, using the tools does not require a mouse; all operations that can be performed are available through menus or keyboard shortcuts. (Menu items that include a shortcut key show that key in the menu.)

### General Options

The following general operations are supported:

- ▶ The left mouse button selects an item.
- ▶ The right mouse button provides a context-sensitive menu for the item.
- ▶ Menu options shown with an ellipsis (...) launch a dialog box.
- ▶ Menu options appear with their associated shortcut keys underlined. (To choose that menu option, press the Alt key + the letter or number underlined in the menu—like Alt+L for “Load State.”)
- ▶ Function keys are assigned to frequently used menu options (for example, the F8 function key starts a build).
- ▶ The use of Arrow and Page keys is supported in the Node list view and tabs (Node Status, Description, File, Size, Conflicts, Messages, and Build Log).

### Context-Sensitive Interactions

Target Wizard supports the following context-sensitive navigation tips:

#### Node List Window

- ▶ Click the arrow next to a node to view (or hide) the components or options in the node.
- ▶ Use the right-arrow key to open a node or move to its first child.
- ▶ Use the left-arrow key to close a node or move to its parent.

- ▶ Right-click a node or select it and press Ctrl+Enter to display its enabling/disabling options.
- ▶ Click the BACK button to return to a dependency in the State information area.

### Dependency Conditions Tab

- ▶ Right-click a dependency or press Ctrl+Enter to display a dependency's enabling/disabling options.
- ▶ Double-click a dependency in the list or highlight a dependency and press Enter to jump to the dependency in the Node list view and resolve the dependency. (Resolving a dependency may require you to enable a node, disable a node, or set a node value.)

## Target Wizard Default Settings

You can set several optional settings in Target Wizard by choosing the File > Settings menu item. The Settings dialog has two tabs: General Options and Project Creation Options.

On the General Options tab, you can set three options:

- ▶ **Show splash screen** - This option controls whether Target Wizard displays the Embedix SDK splash screen while it is starting (Default: Enabled).
- ▶ **Load most recent project** - This option controls whether Target Wizard opens the most recently opened project when you start the program (Default: Enabled).
- ▶ **Save state on exit** - This option controls whether Target Wizard automatically saves the current project before closing the project or exiting Target Wizard (Default: Disabled).

On the Project Creation Options tab, you can set one option: the Default Board that your projects will target. By default, this option is set to Generic x86 board. With each Board Support Package (BSP) that you install in Target Wizard, one board is added to the Default Board selection list on this tab.

Click Default to reset all options to their default settings.



This chapter contains the following major sections:

- ▶ Modifying the Current Project
- ▶ Resolving Package Dependencies and Conflicts
- ▶ Managing Project Size
- ▶ Merging Your Application into the Target Image
- ▶ Converting Your Application into a Custom Embedix Package

## Modifying the Current Project

To customize your project to fit your specific embedded OS requirements, you can selectively add and remove components and functions. This can be done in addition to, or in instead of, loading a configuration profile.

Here is a high-level step-through of the typical customization tasks required of most users.


1. Open your project of choice using one of the following methods:

- ▷ If your project of choice opened by default, go to step 2.

The project most recently worked on is opened by default unless you have changed the settings under the menu item File > Target Wizard Settings.

- ▷ To choose the project from the list of the 10 most recently accessed projects, choose Project > Open Recent Project .
- ▷ To browse the directory structure for the project of your choice, choose Project > Open.

2. Enable or disable nodes (Groups, Components, or Options) or set node values in your project.
  - 2a. Highlight a node (a Group, a Component, or an Option) in the list.
  - 2b. Right-click the node (or press Ctrl+Enter).
  - 2c. Choose an enabling option from the list (such as Disable, Enable, or Enable with Parents).
3. For each node that you enable, check inside the Node Status tab for Dependencies that should also be enabled.
4. Check your Node list (or tree view) for nodes with conflicts.

You must determine which conflicts are acceptable and which are not. For example, for security reasons you may have chosen to enable two different Web servers using two different ports. If so, the system will warn you that you have two Web servers enabled, even though this is part of your project design.
5. Verify that three green checks appear in the Node Status tab by highlighting the Embedix node at the top of the Node list.
  - ▷ If three green checks appear in the Node Status tab when you highlight the Embedix node, then the entire project is enabled, fulfilled, and free of conflicts.
  - ▷ If one or more red Xs appear, you need to determine whether the status is acceptable before you proceed.
6. Save your project. You can choose Project > Save, press Ctrl+S, or simply click the  icon.

## Resolving Package Dependencies and Conflicts

Using a preconfigured state file as a starting point for your project should save you time in fulfilling package dependencies and resolving conflicts. (With a few exceptions where user input is required, the project nodes in state files are fulfilled nodes that are free of conflict.)

However, as you customize your projects, you may cause a node to become unfulfilled by disabling a node you thought you didn't need. Or, you may introduce a conflict by enabling a node that provides a service already provided.

Fortunately, Embedix Target Wizard makes an easy task out of fulfilling dependencies once you are familiar with the icons in the interface. You can quickly recognize the existing of unfulfilled nodes and nodes that are in conflict.

Use the information displayed in the Node Status tab and the Conflicts tab to determine what you need to do to resolve any dependency or conflict. For a detailed example of how this can be done, review the sample project in Appendix A, especially the sections “Resolving Dependencies” on page 66 and “Introducing and Resolving Conflicts into the Tree” on page 70.

## Managing Project Size

Meta-information for a given node, such as the storage size for the group, component, or option, can be seen when you click on the node (or use the Down-arrow key to move to that node).

The information for a particular item is shown in the Size info area (Size tab) on the Embedix Target Wizard screen. You can also see the cumulative total of such information in the *sizes* columns (listed below). This information is shown in two columns in the Node list view. (If you cannot see them, try making the window wider.)

The size information shown for a group or component is the total size of all of the enabled components within that group or component. The available size information includes:

- ▶ Disk Size Estimate
- ▶ Min. RAM Size Estimate

## Reducing Libraries on Your Target

Embedix Target Wizard has a special feature that allows you to reduce the size of the shared libraries on your system to meet your

specific needs. Memory size reduction on most targets can be significant.

For some systems just over a memory boundary (such as 4 MB or 8 MB), this could be the difference between going with a low-cost small memory block rather than a more expensive large one.

You can choose to perform Lipo on your project when you set your project's deployment options. For more information on setting deployment options from the Build > Deploy menu option, see "To build a target project:" on page 38

### **How Lipo Works**

Lipo (the library reduction tool) scans all executables and libraries on the target image, determining which symbols are needed globally by the entire system. Then it searches the libraries, noting symbols that are not used by any executable object. Finally, it constructs new, smaller libraries that include only the needed symbols.

All of this happens after the files in the merge directory are transferred. This ensures that you will have the symbols you need for your custom applications.

### **When Not to Reduce Libraries**

In some cases, you might not want to reduce the libraries.

If you have ample room on your target file system, you probably should not reduce the libraries.

If you are hand-copying executables to the target, you take the risk of not having a symbol you need.

In other words, if you think you might need a symbol in the future that you're not using now, you might want to wait to reduce your libraries.

### **What Happens if Library Reduction Fails?**

When an executable is started, one of the first things that happens is that a program called ld checks the executable for needed libraries and symbols.

If, after scanning through the symbol tree, it finds all symbols defined, it allows your program to start running.

If it finds a symbol missing, it halts the program and notifies you that the particular symbol is missing from the library it is expected to be contained in.

So, if you are missing a symbol, you will find out as soon as you start the application that needs it. As long as you run all your custom binaries, you should be able to detect any missing symbols with ease. (You needn't worry about shipping a product and finding out about a missing symbol from a customer, unless you choose not to do basic testing.)

## Including Symbols Explicitly

It is unlikely that Lipo will remove a needed symbol if you allow Target Wizard to install all files on your target. However, there is a fail-safe mechanism to explicitly include symbols.

If you find you're missing a symbol, you have two good options:

- ▶ Include it in `/tmp/keepsymbols`
- ▶ Turn off library reduction

Put any needed symbols in the file `/tmp/keepsymbols` and the tool will include those symbols and their dependencies in the target libraries. Each symbol requires one line and takes the form "library,symbol."

For example, if you wanted to keep the symbol `malloc` in `libc.so`, you would enter this line:

```
libc, malloc
```

Because library reduction is dynamic, it is impossible to calculate library sizes before the reduction takes place. Therefore, the size calculations shown in Target Wizard may be larger than the actual build. However, because reduced libraries will not be larger than full libraries, you can use the size information as an upper boundary.



**Warning:** Never replace a reduced library with a full library on a target. Doing so will introduce dependencies to symbols that have already been eliminated from other reduced libraries.

---

## Merging Your Application into the Target Image

Merging your custom files into your project image is easy to do by placing the files you want to appear in the image under your project's merge directory. For example, if you had a file called “helloworld” and you wanted it to appear on the target in the path `/usr/local/bin/helloworld`, you would do the following:

1. Place the file in this location:

`<project_dir>/merge/usr/local/bin/helloworld`

If the `/usr/local/bin` directory does not exist in your merge directory, you need to create it.

2. In addition to placing the file into the merge directory, you also need to make sure that you have enabled merging when you run the deployment GUI.



**Note:** Files from the merge directory override those in the image generated by Target Wizard.

---



**Note:** The ownership of the files and directories in the user directory are preserved in the image installed on the target. Therefore, if root needs to own a merged file on the target, make sure that root owns the file in the user directory.

---

Merging directories in order to get your custom files onto a target works well when you have a fixed set of files to place on your target. However, the merging directories method does not provide a way for

you to maintain all configurations under the project state file. It also adds an extra step for you to consider when you build your target.

For additional options on adding a custom application to your project or target image, see the *Embedix SDK Tools* book.

## Converting Your Application into a Custom Embedix Package

An alternative method of adding a custom application to the target image is to “package” your custom files or application and then add your custom package to your Target Wizard project so that it will build alongside the Embedix SDK packages.

This is easily done with the use of the Package Editor (available from the Target Wizard Tools menu and from the command line).

For detailed information on the Package Editor and how to use it, refer to the *Embedix SDK Tools* book.





## Building and Deploying a Target Image

---

Once you have created and customized your Embedix® Target Wizard project, you can build your target image and deploy it to your target. To assist you in this process, see the following sections of this documents:

- ▶ “Building a Target Project” on page 37
- ▶ “Changing Project Build Options” on page 39
- ▶ “Deploying a Target Image—General Guidelines” on page 39
- ▶ “Deploying a Target Image to an x86 Target” on page 40
- ▶ “Deploying to and from a VM-ware Hosted Virtual Linux Machine (SDK for Windows Users Only)” on page 50



---

**Note:** For detailed instructions on deploying to a specific board, refer to the documentation that came with the BSP of your choice.



---

### Building a Target Project

After you have customized your project and verified that the packages you need are enabled and fulfilled, and that the project is free of unacceptable conflicts, your project is ready to be built.

Embedix Target Wizard allows you to build and rebuild Embedix components as needed. You can choose to only build those components that have had a configuration change since the last build. Or, you can choose to force a rebuild of all of the included components (even those that have not changed).

### To build a target project:

1. If you haven't already done so, from the shell prompt, start Embedix Target Wizard by entering the command **tw**.
2. If your project of choice doesn't open by default, from the menu bar, choose **Project > Open Recent** or **Project > Open** and browse for and select the project.
3. From the menu bar, choose either **Build > Build** or **Build > Force Rebuild**.
  - ▷ To do either an initial build or an incremental build on your project, from the menu bar, choose **Build > Build** (or press F8 or click the  icon). This will build only packages that have changed since the last build and will include Build Options that are checked.
  - ▷ To rebuild the entire project, from the menu bar, choose **Build > Force Rebuild** (or click the  icon). This will build all packages (regardless of whether or not they have changed since the last build) and will include Build Options that are enabled.

When you do a build, the resulting components (RPM files) are placed in the directory `<project>/build/rpmdir/RPMS/i386`. This becomes the reference directory for subsequent builds. If the timestamps match, the component is not rebuilt, which saves you time in the end.

If you change the location of the default *project* directory and attempt to build the Embedix components, Target Wizard attempts to do a complete rebuild of the project.

Embedix Target Wizard manages rebuilds of the Linux kernel itself in a slightly different way. Although the kernel is rebuilt from source code, the decision to rebuild is based on whether changes were made to the kernel configuration. If the configuration does not change, the kernel is not rebuilt.

All packages are built in the Build Directory. The resulting kernel image is left in the project directory.

## Changing Project Build Options

When you create a project, you enable the initial Build Options for the Target Image (such as Build image after packages or Merge user and target images). For any existing project, you can change the enabled build options by completing the following steps:



---

**Note:** For a complete list of options and descriptions, see “Creating a Project” on page 9.

---

1. If you haven't already done so, from the shell prompt, start Embedix Target Wizard by entering the command `tw`.
2. If your project of choice doesn't open by default, from the menu bar, choose **Project > Open Recent** (or **Project > Open**) and then browse for and select the project.
3. From the menu bar, choose **Project > Options**.
4. Click each option you want to enable or disable, then click OK.

## Deploying a Target Image—General Guidelines

Deployment instructions and options are target platform-specific. Where specific target deployment instructions exist, those instructions supersede any general information contained in this document.

**Deploying to Board Support Package (BSP) Board:** For instructions on deploying to a BSP board, refer to the documentation that shipped with your BSP.

**Deploying to an x86 Target:** Most developers will find the x86 deployment instruction helpful at some point, whether they plan to do x86 deployments or not. Refer to “Deploying a Target Image to an x86 Target” on page 40.

**Deploying to a Virtual Linux Machine:** If you are using the VM-ware hosted virtual Linux machine that ships with Embedix SDK for

Windows, refer to “Deploying to and from a VM-ware Hosted Virtual Linux Machine (SDK for Windows Users Only)” on page 50.

## Deploying a Target Image to an x86 Target

No matter which board support package (BSP) shipped with your Embedix SDK, you were provided several default deployment options for x86 targets.

All of the x86 deployment options begin by starting the Deployment Wizard. Complete the steps listed in “Start the x86 Deployment Wizard” on page 40 and refer to the instructions for your chosen deployment option.

### Start the x86 Deployment Wizard

1. If you haven't already done so, open your x86 (target platform) project of choice and build your target project.
2. From the menu bar, choose Build > Deploy (or from the tool bar, click on the Deploy button).
3. At the Welcome screen, make sure that the project directory is correct and click NEXT.
4. Enter the path to the kernel image and click NEXT.

A default is provided. If you do change the path, note that, as a safeguard, the kernel image must have the project directory in its path. You may enter a path or browse to one in your file system.

5. At the Build Filesystem page, select all, some, or none of the three build options and click NEXT:
  - ▷ **Rebuild Filesystem:** This creates the directory tree from the tar files built by Target Wizard
  - ▷ **Add Merge Directory Contents:** This adds the contents of the merge directory to the directory tree
  - ▷ **Add Kernel to /boot:** This will place the kernel in the boot directory of the directory tree

Most users will need to select at least Rebuild Filesystem.

If you select Rebuild Filesystem or Add Merge Directory Contents, then the Build Log screen appears, which contains a list of all of the files extracted and placed into the directory tree.

6. If the Build Log screen appears, ensure that there are no errors in the log and click NEXT.

When errors are present, some of the required files will be excluded from the build.

7. (Optional) At the Running Lipo screen, enable Run Lipo and click NEXT to display the list of libraries that are not used by any executables on the target file system.

Lipo is a utility used to reduce the size of libraries on the target file system. You most likely won't need to run Lipo during the development phase, but you can choose to run it anytime. For more information on Lipo, see "Reducing Libraries on Your Target" on page 31.

If you don't want to run Lipo at this time, just click NEXT.

8. (Optional) If you ran Lipo and a list of unused libraries displayed, highlight the libraries you want to delete and click NEXT.
9. At the Deployment Options screen, choose a deployment method, click NEXT and refer to the option-specific instructions to help you complete your deployment.

"Make a Directory Tree" on page 44

"Make a Self-Hosting RAM Disk Filesystem" on page 44

"Make a Filesystem to chroot Into" on page 45

"Make Install Disks for a RAM Disk System" on page 45

"Make Install Disks for a Normal Filesystem" on page 47

"Make a Bootable Install CD" on page 48

"Make a CD that Boots to the OS" on page 49

For more information about deployment options, see Table 6-1.



**Note:** The browse sequence of the deployment wizard is displayed in the left pane of the Deployment Wizard window. You can customize the Deployment Wizard by editing the deployment script, which is located at: `<projectdir>/emb-bin/<target_platform>_emb_deploy`. For example, if your chosen target platform is `Generic_x86_board`, then the filename would be `Generic_x86_board_emb_deploy`.

---

## x86 Deployment Options

Table 6-1 describes the x86 deployment options available in this release of Embedix SDK.



**Note:** Deployment instructions and options are target platform-specific. Where specific target deployment instructions exist (such as in an Embedix SDK board support package), those instructions supersede any general information contained in this document.

---

**Table 6-1.** x86 Deployment Options

Deployment Options	Description	Requirements	Why you would use it
“Make a Directory Tree” on page 44	Creates a directory tree of the target file system in a directory of the users choice	None	Check to make sure that all files are in the right place, if you have another machine mounted somewhere in your file system
“Make a Self-Hosting RAM Disk Filesystem” on page 44	Creates a RAM disk image and adds an entry to the bootloader on the host system to allow booting into the RAM disk image with the built kernel	Kernel must have RAM disk support built in. Host machine must have enough physical ram to hold the file system.	If you have no other target machine and would like to make sure that the kernel works and the file system is complete

**Table 6-1.** x86 Deployment Options

Deployment Options	Description	Requirements	Why you would use it
“Make a Filesystem to chroot Into” on page 45	This places a copy of the directory tree in a directory of the user's choice and gives instructions on how to chroot into it	None	Allows you to test the file system and run programs as if you were really on the target. No need to worry about the kernel or booting.
“Make Install Disks for a RAM Disk System” on page 45	This creates a boot install floppy disk and a multi-volume tar set of floppies. It installs a kernel and a RAM disk image onto a partition on the target machine.	Kernel must have ram disk support built in. Enough physical memory on target to hold the entire file system.	Allows you to test the kernel and file system on the target. May be used as a production deployment method. No persistence of data.
“Make Install Disks for a Normal Filesystem” on page 47	Creates a boot install floppy and a series of data floppy disks. The install copies the kernel and an ext2 file system to a partition on the target and sets up the bootloader.	Kernel must support ext2 file system and ide block devices	Allows you to test the kernel and file system on the target. May be used as a production deployment method. Changes made to the file system persist since the file system is not a RAM disk.
“Make a Bootable Install CD” on page 48	This is similar to making install disks for a normal install. This creates a bootable cd ISO image. The image can be burned onto a cd and the cd can be used to install the kernel and (ext2) file system onto a partition of the target.	The kernel must support ide block devices (or scsi if the file system is put onto a scsi device)	Allows same flexibility as Make install disks for normal file system, but the install is much faster. Useful if the target file system is large or you plan on deploying to many machines.

**Table 6-1.** x86 Deployment Options

Deployment Options	Description	Requirements	Why you would use it
“Make a CD that Boots to the OS” on page 49	Creates a bootable iso cd image that boots using the built kernel into the target file system on a RAM disk (from the cd)	Kernel must support RAM disk, iso 9660 Filesystem, and have CD-ROM ide (or scsi for a scsi drive) device support. target machine must have enough physical ram for the target file system.	Allows you to carry a whole operating system on CD and boot on any machine without touching the hard drive. No hard drive needed on target machine. No persistence of data.

---

## Make a Directory Tree

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. In the text box, enter the directory where you would like the directory tree to be place and click NEXT.  

You may look at your file system by clicking on the browse button. If you enter a directory that does not exist, it will be created for you.
3. At the Instructions screen, check where the directory tree was placed and click FINISH.

## Make a Self-Hosting RAM Disk Filesystem

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.  

The text box will display the entry that you need to add to your lilo.conf file in order to boot into a RAM disk file system with your kernel.
2. Open `/etc/lilo.conf` and add the entry to the end of it.



3. Set Ram Disk Size, ensuring it is large enough to hold the file system and not larger than the amount of physical ram on your machine, and click NEXT.
4. Exit the wizard and close all applications on the host machine.
5. Reboot your machine and select the label that you entered in lilo.conf file.

### Make a Filesystem to chroot Into

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. In the text box, enter the directory where you would like the directory tree to be place and click NEXT.  

You may look at your file system by clicking on the browse button. If you enter a directory that does not exist, it will be created for you.
3. Read the instructions on the last page of the wizard.
4. Click the Terminal Emulation (X-term) icon on your screen to access a shell prompt.
5. At the shell prompt, enter: `chroot <dir> /bin/sh` where `<dir>` is the directory that you specified.
6. At the shell prompt, enter: `cd /` to get to the root of the file system.

### Make Install Disks for a RAM Disk System

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. On the Target Partition screen, highlight the partition on the target machine that you want to deploy to and click NEXT.  

You will have a chance to override this later if needed.
3. On the Make Install Floppy screen:
  - 3a. (Optional) Keep the Make Install Floppy option enabled to prepare to make an install floppy.

You may decide to bypass making the install floppy disk, but it is not recommended.

- 3b.** Set Ram Disk Size, ensuring it is large enough to hold the file system and not larger than the amount of physical ram on your machine.
- 3c.** Click NEXT.
- 4.** If you kept the Make Install Floppy box enabled, then, when prompted to do so, place a floppy disk labeled "install floppy" in the drive and click NEXT.
- 5.** On the Make RAM disk Floppy Disks screen:
  - 5a.** (Optional) Keep the Make Target Floppies option enabled.

This will copy the kernel and file system onto a series of floppy disks. You may bypass this step if you already have a set of floppies, but it is not recommended.
  - 5b.** Click NEXT.
- 6.** If you kept Make Target Floppies enabled, then, when prompted to do so, label a floppy disk, insert it into the drive, then click OK.

You may need to repeat this step with several floppy disks, depending on the size of the file system and kernel.
- 7.** Read the instructions on the last page of the wizard.
- 8.** To run the install script, place the install floppy in the floppy drive of the target machine and boot it.
- 9.** Enter the partition to deploy to.
- 10.** If the partition is ready, press Y and Enter and follow the screen prompts, inserting RAM disk floppy disks as needed.

If the partition is not ready, press N and Enter, then run fdisk on the partition to set it up, and finally run the install script again by entering **./install**
- 11.** When prompted to do so, choose Yes to install the boot loader.
- 12.** If this is the only operating system on the target machine, then do one of the following:

- ▷ Install the boot loader onto the master boot record (e.g./dev/hda).
  - ▷ Place the boot loader on the partition (e.g. /dev/hda1).
13. Remove the floppy disk from the drive and press Ctrl+Alt+Delete to reboot the system.

## Make Install Disks for a Normal Filesystem

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. On the Target Partition screen, highlight the partition on the target machine that you want to deploy to and click NEXT.  
You will have a chance to override this later if needed.
3. On the Make Install Floppy screen:
  - 3a. (Optional) Keep the Make Install Floppy option enabled to prepare to make an install floppy.  
You may decide to bypass making the install floppy disk, but it is not recommended.
  - 3b. Click NEXT.
4. If you kept the Make Install Floppy box enabled, then, when prompted to do so, place a floppy disk labeled "install floppy" in the drive and click NEXT.
5. On the Make Normal FS Floppy Disks screen:
  - 5a. (Optional) Keep the Make Normal Target Floppies option enabled.  
This will copy the kernel and file system onto a series of floppy disks. You may bypass this step if you already have a set of floppies, but it is not recommended.
  - 5b. Click NEXT.
6. If you kept Make Normal Target Floppies enabled, then, when prompted to do so, label a floppy disk, insert it into the drive, then click OK.

You may need to repeat this step with several floppy disks, depending on the size of the file system and kernel.

7. Read the instructions on the last page of the wizard.
8. To run the install script, place the install floppy in the floppy drive of the target machine and boot it.
9. Enter the partition to deploy to.
10. If the partition is ready, press Y and Enter and follow the screen prompts, inserting file system floppy disks as needed.  
If the partition is not ready, press N and Enter, then run `fdisk` on the partition to set it up, and finally run the install script again by entering `./install`
11. When prompted to do so, choose Yes to install the boot loader.
12. If this is the only operating system on the target machine, then do one of the following:
  - ▷ Install the boot loader onto the master boot record (e.g./dev/hda).
  - ▷ Place the boot loader on the partition (e.g. /dev/hda1).
13. Remove the floppy from the drive and press Ctrl+Alt+Delete to reboot the system.

### Make a Bootable Install CD

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. On the Target Partition screen, highlight the partition on the target machine that you want to deploy to and click NEXT.  
You will have a chance to override this later if needed.
3. On the Make Bootable Install CD screen, keep Make CD Image enabled and click NEXT.  
A bootable iso image named `bootcd.iso` is created in the `tmp` directory of the project directory.
4. Burn the iso image to a CD.

5. Read the instructions on the last page of the wizard.
6. To run the install script, make sure that the bios of the target machine allows CD booting, then place the CD in the target machine's CD-ROM drive and boot it.
7. Enter the partition to deploy to.
8. If the partition is ready, press Y and Enter and follow the screen prompts.  
  
If the partition is not ready, press N and Enter, then run fdisk on the partition to set it up, and finally run the install script again by entering `./install`
9. When prompted to do so, choose Yes to install the boot loader.
10. If this is the only operating system on the target machine, then do one of the following:
  - ▷ Install the boot loader onto the master boot record (e.g./dev/hda).
  - ▷ Place the boot loader on the partition (e.g. /dev/hda1).
11. Remove the CD from the drive and press Ctrl+Alt+Delete to reboot the system.

## Make a CD that Boots to the OS

1. Complete the steps outlined in “Start the x86 Deployment Wizard” on page 40.
2. On the Make Bootable OS CD screen:
  - 2a. Keep Make CD Image enabled to prepare to make a bootable iso image.
  - 2b. Set Ram Disk Size, ensuring it is large enough to hold the file system and not larger than the amount of physical ram on your machine.
  - 2c. Click NEXT.  
  
A bootable iso image named bootcd.iso is created in the tmp directory of the project directory.

3. Burn the iso image to a CD.
4. Read the instructions on the last screen of the wizard.
5. To run the install script, make sure that the bios of the target machine allows CD booting, then place the CD in the target machine's CD-ROM drive and boot it.
6. At the console prompt, enter `./install` to populate the RAM disk with the target file system.

## Deploying to and from a VM-ware Hosted Virtual Linux Machine (SDK for Windows Users Only)

If you are an Embedix SDK for Windows user and are deploying to an x86 target, several deployment options are available, including installation floppy disks and compact disks.

- ▶ If you have a target device (or board) connected to your physical workstation, you may be able to deploy to it directly from the virtual machine.
- ▶ If you want to create installation floppy disks for your target, use VMware's Devices menu to verify that the floppy disk drive is connected to your virtual machine.
- ▶ If you want to transfer a target image between the virtual machine and the file system of a Windows workstation, use the virtual machine's FTP server to transfer the file(s).

For complete information about supported deployment methods, refer to your Embedix SDK Board Support Package documentation in addition to the complete contents of this chapter. The information in this section supplements these documents and focuses primarily on what is unique about the following deployment methods when deploying from a virtual machine:

- ▶ "Deploying to an External Hardware Target" on page 51
- ▶ "Deploying using Floppy Disks" on page 51
- ▶ "Sharing Files with the Windows File System" on page 52

## Deploying to a External Hardware Target

You can deploy Embedix images to external targets using standard connections.

- ▶ Many external targets accept TCP/IP connections and can communicate with the virtual machine using FTP and NFS protocols.
- ▶ Some target boards use serial connections to communicate with the host.

If you are deploying to one of the many boards that have an Embedix SDK Board Support Package (BSP) available, refer to your BSP documentation for board-specific deployment options.

For general information about deploying information to external hardware targets, see the *Target Wizard User Guide*.

## Deploying using Floppy Disks

You can deploy a target image to floppy disks, which you can then use to install your custom Embedix operating system on another device.

Before you begin deploying a target image from the virtual machine to diskette, use the VMware Tools to verify that your floppy disk drive is connected to your virtual machine. For more information about connecting devices to the virtual machine, refer to “Using VMware Tools for Linux” on page 29.

For more information about deploying your target image using floppy disks, refer to the *Target Wizard User Guide*.



---

**Warning:** Embedix SDK for Windows does not support deployment of an Embedix target image on your Windows workstation, either in a virtual machine or on the workstation file system. Use a separate x86-based computer to deploy and evaluate any x86 target images.

---

## Sharing Files with the Windows File System

The virtual machine includes a pre-installed FTP server, which requires that networking is enabled in VMware.

You can use a Windows FTP client to securely transfer files to and from the virtual machine, as discussed in the two following sections:

- ▶ Copying Files to the Virtual Machine's FTP Server
- ▶ Connecting to the Virtual Machine's FTP Server

### Copying Files to the Virtual Machine's FTP Server

Before you can use the virtual machine's FTP server, you need to copy or move the files to the FTP server's `/pub` directory. You can transfer files and directories to and from the server using standard FTP commands, browsers, and other utilities.

In the following example, the file `bootcd.iso` is moved from `/home/vkit/project/embedix/tmp` to `/home/ftp/pub`.

```
mv /home/vkit/project/embedix/tmp/bootcd.iso /home/ftp/pub
```

To copy all contents of the `tmp/` folder, including any directories, use the following command:

```
cp -a /home/vkit/project/embedix/tmp /home/ftp/pub
```

For more information about Linux commands and their syntax, refer to the Linux documentation.



---

**Warning:** Any files and subdirectories in the folder `/home/ftp/pub` are available to your Windows workstation and other FTP clients on your network.

---

### Connecting to the Virtual Machine's FTP Server

Before you can connect to the virtual machine's FTP server, you need to know its IP address.

- ▶ To determine the virtual machine's IP address, open a Linux terminal window and enter the following command:

```
/sbin/ifconfig eth0
```



The virtual machine's IP address is listed on the second line of the system's response, prefaced by the string `inet addr`.

- ▶ To copy files from the virtual machine's FTP server to a Windows workstation, invoke FTP from the Windows command prompt or use an FTP utility, such as an Internet browser.

For instance, from Netscape Navigator on your Windows workstation, you can use the following procedure to retrieve a file named `bootcd.iso`:

1. To display available files, enter the following command in the browser's `Location` field:

```
ftp://a.b.c.d/pub
```

where *a*, *b*, *c*, and *d* are one- to three-digit elements of the FTP servers's IP address.

2. Right-click the `bootcd.iso` file and then choose the menu item `Save Link As`.
3. Save the file in an appropriate location in your Windows workstation's file system.

From Internet Explorer, replace `Save Link As` with the `Copy to Folder` command.

Deploying to and from a VM-ware Hosted Virtual Linux Machine (SDK for Windows Users)

# A Exploring a Sample Project

---

The following section describes the creation and exploration of a sample Target Wizard project.

Every Target Wizard project is host development machine-specific (Linux or Windows) and target platform specific (such as ARM, PowerPC, or x86). However, regardless of your host or target system, you may find this sample project helpful. While your tree view and menus may vary, the basic principles are the same.

This section includes the following subsections.

- “About the Host and Target” on page 55
- “Creating the Sample Project” on page 56
- “Choosing and Customizing a Configuration Profile” on page 62
- “Exploring the Tree” on page 65
- “Resolving Dependencies” on page 66
- “Introducing and Resolving Conflicts into the Tree” on page 70
- “Checking Project Status” on page 73

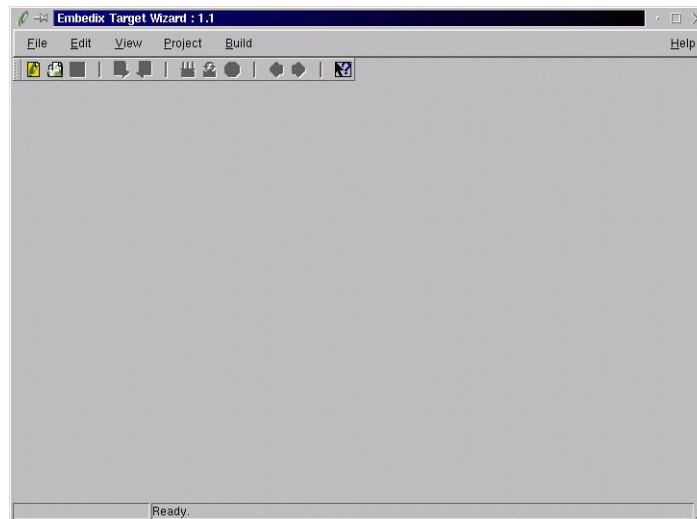
## About the Host and Target

For this sample project, we used the following setup:

- ▶ Host platform = Linux development machine
- ▶ Target platform = Generic x86 board

## Creating the Sample Project

The first time you launch Embedix Target Wizard, the following screen displays.

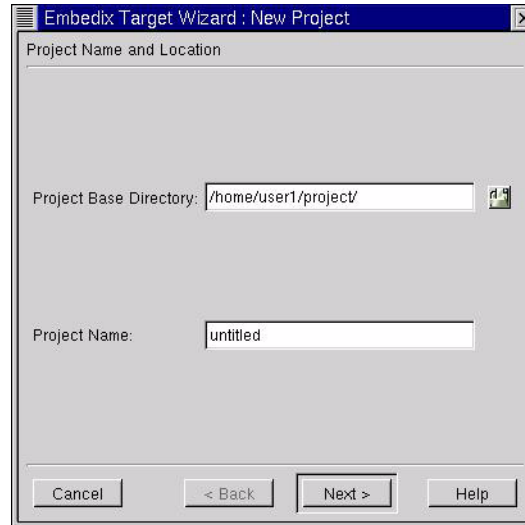


Because no project has been created yet, the main viewing window is empty and other screen elements do not display.

When you create a new project, you must specify its name, working directories, and build options. When this is completed, the screen will be populated. To create a new project,

1. Choose Project > New.

The *Project Name and Location* dialog box displays.



2. Enter a Project Base Directory and a Project Name.

**Project Base Directory** is the parent directory where your project-specific directory will reside.

**Project Name** is the name you want to give this new project.

The project name becomes a directory name, so only the following characters are valid characters for your project name:

[A-Z], [a-z], [0-9], [·], [·], [-]

The program will automatically create a directory with the same name as your project in the Project Base Directory unless the directory already exists.

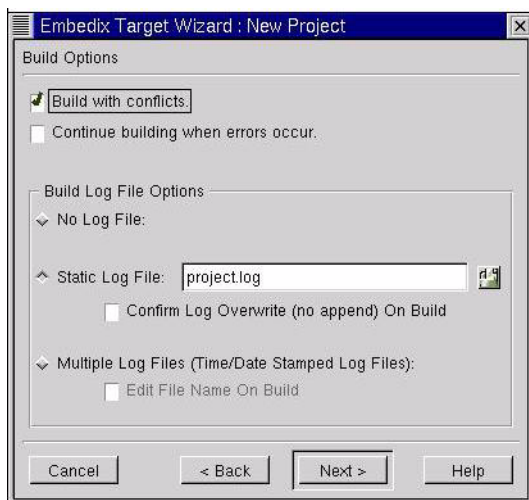
For example, if you accepted the default Project Base Directory, `/home/<username>/project` and then entered the Project Name `x86Test`, the system would create the directory: `/home/<username>/project/x86Test`.



**Note:** The project directory is created at the location specified, but during project creation, the actual project files are moved to a directory under /opt/Embedix, and the project directory is replaced with a symlink.

---

3. When you have completed this first dialog box, click NEXT. The Build Options screen displays.



4. You can select one or more build options for this project. You can also change the Log File options.

**Build with conflicts** allows builds to continue when conflicts exist.

**Continue building when errors occur** allows builds to continue when errors occur.

**Build Log File Options** allow you to choose the type of log file you prefer. Here are your choices and the results:

- ▷ **No Log File:** No log file produced, but you can still view the last 200 lines of build history from your project's Build Log tab.

- ▷ **Static Log File:** The log file specified in the entry box will be appended to with each build unless you select the “Confirm Log Overwrite” option. If you select this option, a dialog box will open with each build to allow you to select one of the following log file options: Append, Overwrite, or Rename. The log file will reside in the current project directory unless you enter a full path to another directory.
- ▷ **Multiple Log Files:** A new log file is generated in the current project directory for each build with a date and time stamp in the filename. The log file name format is:  
*project name\_year\_month\_day\_hour.minute.second.logfile.*  
For example, if you built x86Test with this option selected just before 9:30 in the morning, the system would generate the log file: x86Test\_2001\_5\_11\_9.29.59.logfile



**Note:** No matter what log file option you choose, the Build Log tab will always display the last 200 lines of build output for the last build initiated.

---

5. When you have set your Build Options, click NEXT.

The Target Options screen displays.

The options provided in the drop-down list will vary depending on which board support packages (BSPs) you have installed.

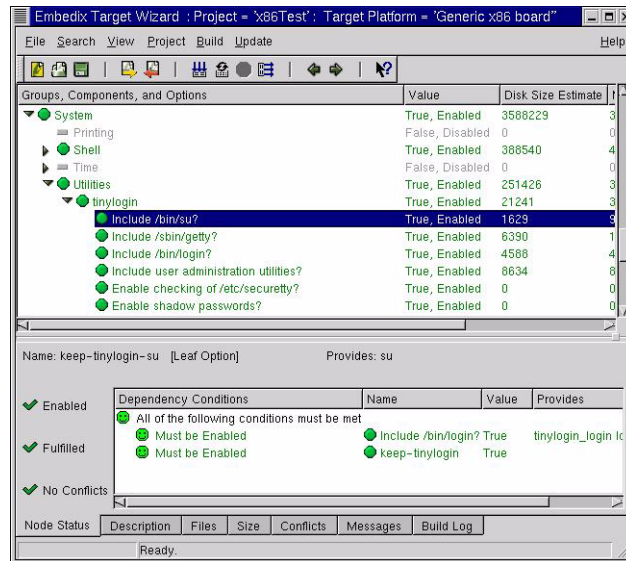


6. From the drop-down list, choose the board support package *Generic x86 board* as your target platform for this exercise, and then click FINISH.

The system takes a few minutes to create your project.

When the system returns you to the Embedix Target Wizard interface, the screen is populated by SDK packages, which display by default in package groups in the Groups, Components, and Options viewing window (also known as the “tree” view).







In this edition of Embedix SDK, every new project begins as a preconfigured, prebuilt project for the target platform you choose in Step 6. Later, you can choose from among several other configuration profiles (or states) provided by Lineo to save you setup time and build time.

If you had chosen a BSP platform as your target platform, then, by default, you would have been given a default configuration appropriate for that platform.

But because you chose *Generic x86 board* as your target platform, then, by default, you were given a configuration appropriate for i386 projects.

The viewing window (or tree view) contains the package groups view of your project, with groups that are  enabled and  disabled. (For a detailed list of the packages in each group and the full path to each package, see “Package Groups in Target Wizard” on page 82.)



**Note:** To view your project nodes by component (or package) rather than groups, choose View > Show Groups to disable that option.

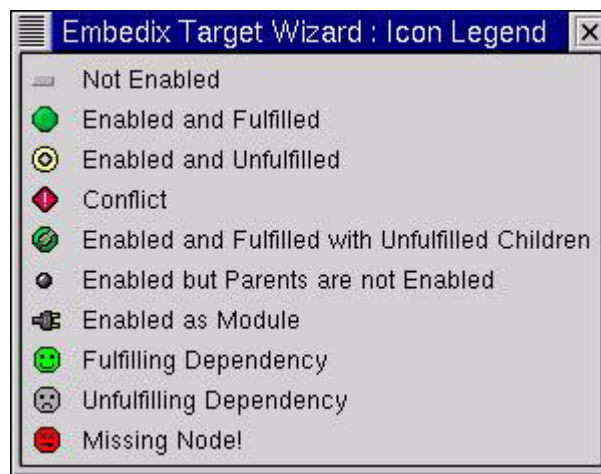
---



**Tip:** For a quick reference of the Embedix Target Wizard icons as you explore the interface, see **Icon Legend** under the Help menu. When you choose Help > Icon Legend, the legend (see Figure A-1) remains on top of the other opened screens until you close it.

---

**Figure A-1.** Icon Legend



Continue to the next section to learn more about loading configuration profiles that ship with Embedix SDK into your project.

## Choosing and Customizing a Configuration Profile

Embedix Target Wizard ships with several configuration profiles (or collections of preselected system settings). Most users find it useful to start a new project with a configuration profile, so the program has been designed to select your project's default configuration profile

based on your choice of target platform in the New Project wizard (Step 6 on page 60).

Before completing any other system customization, you can choose to load a different configuration profile (or state) into your project and customize it to meet your specific need.

1. (Optional) To choose a different configuration profile, choose File > Load State, and then choose a state from the menu.

For *Generic x86 board* target platform projects, the following configuration profiles or “states” are available in this product:

- ▷ **i386\_medium**  
Embedix Linux configuration settings preselected by Lineo to meet x86 embedded Linux requirements for 26 packages that will fit on a 16 MB RAM disk.
- ▷ **i386\_many**  
Embedix Linux configuration settings preselected by Lineo to meet x86 embedded Linux requirements for 47 packages, which is a good option for build testing.
- ▷ **Embedix\_min**  
Minimal Embedix Linux configuration setting preselected by Lineo to meet tighter size requirements for an x86 embedded Linux. This simply provides the setup for a system with a shell.
- ▷ **Embedix\_config\_1.0**  
Embedix Linux configuration settings preselected by Lineo to meet typical x86 embedded Linux requirements.
- ▷ **settop\_config\_pc**  
Embedix Linux configuration settings preselected by Lineo to meet typical x86 embedded Linux requirements on a set-top box.

If you have installed one or more Embedix BSPs (board support packages that are Embedix SDK plugins) and create a BSP board target platform project, you are provided one or more board-specific states to choose from.

You can also save any project's configuration (or a portion of a project's configuration) as a state file and have it display as another option in your File > Load State list (described in Step 4 on page 64).

2. After loading a configuration profile, explore the tree and determine if you need to configure any settings manually to meet your particular needs.
3. Edit your project settings as needed.

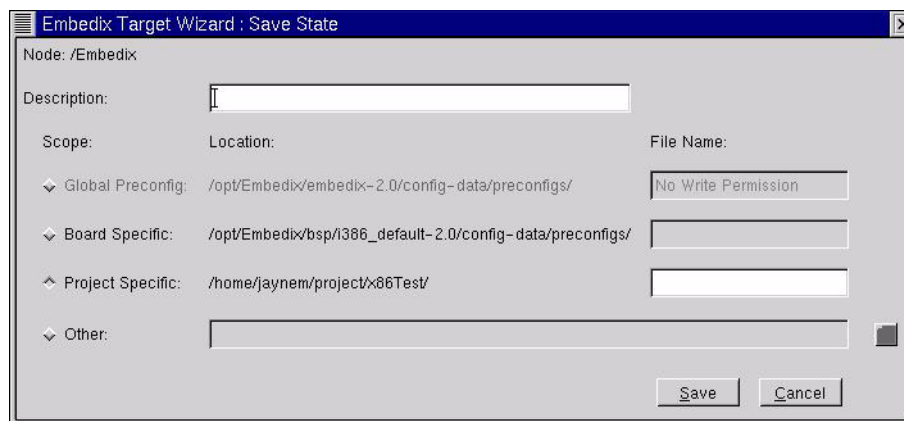


**Warning:** Some “parent” nodes may display as fulfilled, even in the absence of an enabled child. Ensure that all specific “child” nodes that you need are enabled before selecting Project > Save.

---

4. (Optional) After customizing your project's configuration, you may want to save the project configuration (or a portion of the project configuration) as a configuration profile or state file (.sdf) so that you can reuse your custom settings again and again for other projects.
  - 4a. Choose File > Save State for Entire Project to save the configuration settings of your entire current project (or choose File > Save State for Current Node to save the configuration settings of just the node you have highlighted and its children).

The following dialog box appears.



- 4b. Choose one type of state file (Global Preconfig, Board Specific, Project Specific, or Other), complete the dialog box, and then click SAVE.

Your custom state file should now appear in the State Files list when you choose File > Load State.

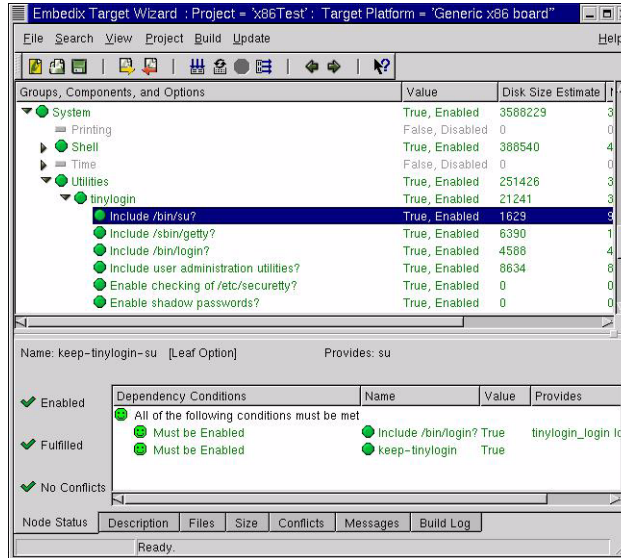
## Exploring the Tree

To explore the contents of the System group,

1. Expand the tree view of the System group by clicking the node with mouse (or by arrowing down and using the Right-arrow on the keyboard).
2. In the System group, move down through **Utilities** and **tinylogin** in order to view the **Include /bin/su** (super user) node.


The results are shown in the following screen shot.

**Include /bin/su** is enabled by default as part of the configuration profile you loaded.



Notice the Dependency Conditions list that displays in the Node Status tab (in the lower part of the screen). These items indicate that the **Include /bin/su** node has dependencies on both **Include /bin/login** and **keep-tinylogin**.

Both are marked “Must be Enabled,” which means that both **Include /bin/login** and **keep-tinylogin** must be enabled if **Include /bin/su** node is to be enabled and fulfilled.

 The *Fulfilling* (green happy face) icon indicates that the item it appears next to is helping to fulfill the node that is highlighted in the Node List view.

## Resolving Dependencies

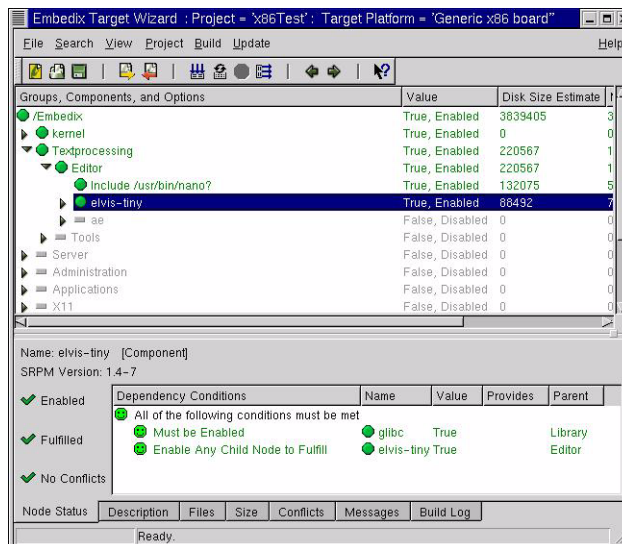
Dependencies that need to be met will exist in every project. Fortunately, Embedix Target Wizard makes an easy task out of fulfilling dependencies.




**Note:** You can save your updates to this project at any time by choosing Project > Save, or you can continue fulfilling dependencies and resolving conflicts until you are ready to build.

Using the sample project you created earlier (see page 56), continue exploring Embedix Target Wizard.

1. In the tree view, go to Embedix > Textprocessing > Editor > elvis-tiny and then review the Dependency Conditions in the Node Status tab.

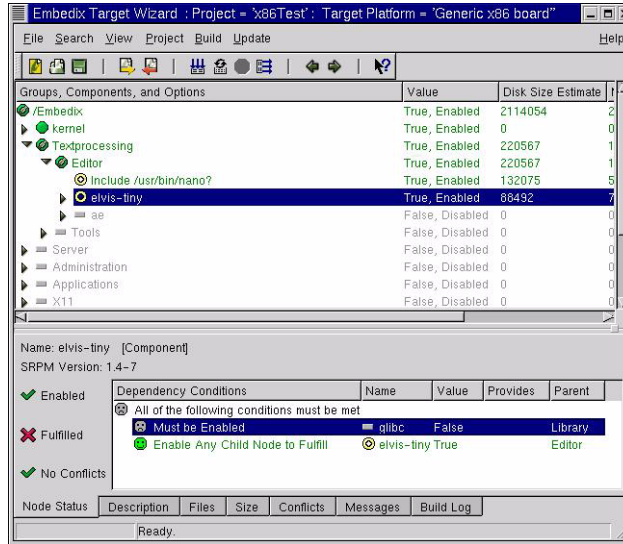


Notice that dependencies are grouped under the instruction or category “All of the following conditions must be met.” Notice also that one of the dependency conditions for elvis-tiny is that glibc “Must be Enabled.”

Since all of the dependencies have been met, glibc displays the  “Enabled and Fulfilled” icon.

See what happens when you disable glibc.

- Right-click on the glibc list item (“Must be Enabled”) and then choose Disable.

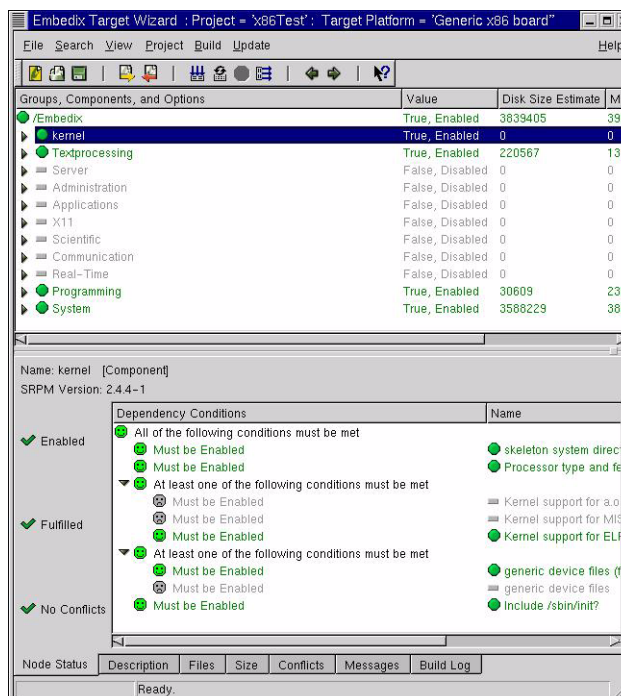


Notice the new “Enabled and Unfulfilled” icon next to elvis-tiny node. Notice also the trail of parent nodes with the “Enabled and Fulfilled with Unfulfilled Children” icon next to them. The trail leads all the way back to Embedix.

Following the icon trail works both ways. You can also start at the top of the tree and look in the Embedix groups for unusual icons and work your way down to a potential problem area.

- Right-click again on the glibc list item and then choose Enable with Parents to return to once again enable and fulfilled the elvis-tiny node.
- Highlight the kernel group and review its Dependency Conditions list in the Node Status tab.





Note that these dependencies are grouped under two different instructions or categories:

“All of the following conditions must be met”

“At least one of the following conditions must be met”

For this highlighted node, all of the requirements have been met even though some of the list items are not enabled.

Notice that all dependencies in the “All of the following...” list appear with happy faces. This means that this group has met the dependencies, which in turn fulfills the Include...dhcpcd item and all of its upline parents in the hierarchical view.

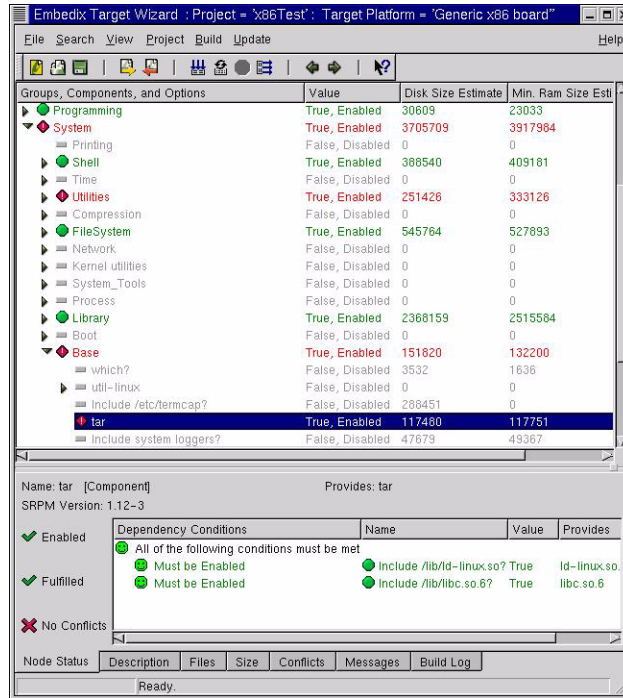
When all dependency items under the All of the following... list are met, the icon that appears next to the list changes to a green Happy Face and the Include...dhcpcd icon indicates that it is now fulfilled.


Be aware that if you enable more than one item in the At least one of the following... list, you are likely to introduce a conflict. On the other hand, if you fail to meet all dependencies listed in the All of the following... list, you will not help fulfill the highlighted node.

## Introducing and Resolving Conflicts into the Tree

As you enable different nodes in your project, you will likely introduce conflicts between enabled nodes. To see an example of this, enable another component in the System group .

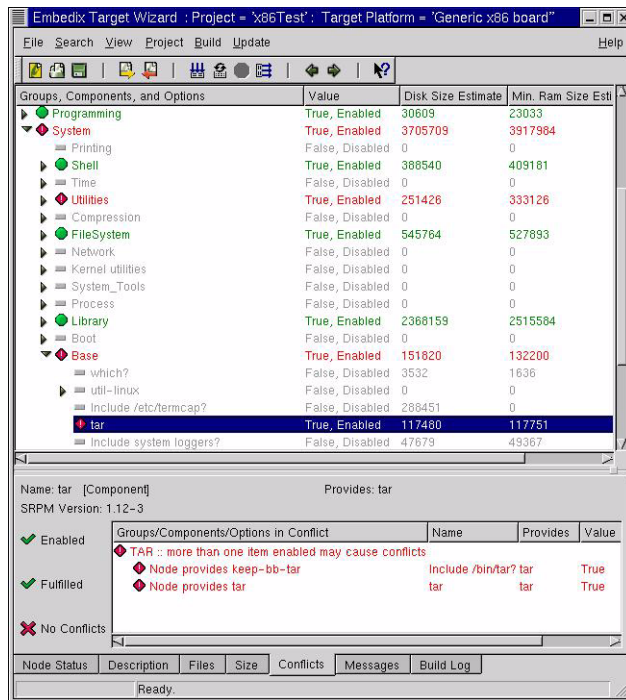
1. Move down the hierarchical (or tree) view to System > Base, and then enable **tar**.



The  *Conflict* icon now appears next to **tar** and a few other nodes. This means that either this node is in conflict with an enabled node or this node contains conflicting enabled nodes.

You need to explore and resolve this conflict, if possible.

- To determine how to resolve this conflict, with **tar** still highlighted, click the Conflicts tab (in the lower part of the screen).



The list displayed in the Conflicts tab (“Group/Components/Options in Conflict”) explains the source of the conflict. In this case, two different enabled components are providing **tar**—*busybox* and *tar*.

To resolve the conflict, you must disable one of these tar nodes.




**Tip:** When deciding which conflicting node to disable, it’s important to know that a node contained in **busybox** is

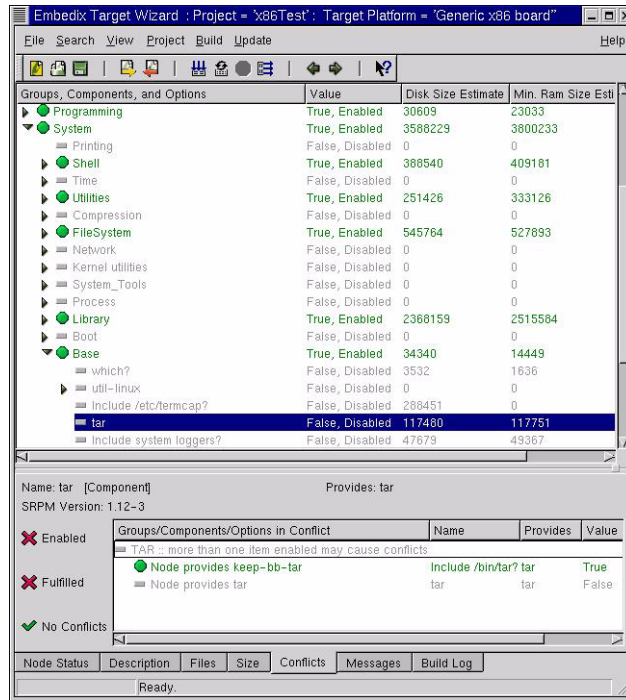
generally the smaller version. So, in the absence of other considerations (such as a dependency on another version), keep the **busybox** version enabled and disable the other.

3. Disable the non-busybox provider of tar by right-clicking on the “Node provides tar” node inside the Conflicts tab and choosing “Disable conflict node.”



**Note:** You can also locate existing conflicts by exploring the tree view. If a child node is in conflict with another node, the parent node is also marked with a  Conflicts icon.

Once you resolve the conflict, all Conflict icons are replaced by Enabled and Fulfilled icons.



## Checking Project Status

To do a quick check of the state of your project,

1. At the top of the tree view, highlight Embedix and then choose Project > Save.
2. Determine whether three green checks appear in the Node Status tab.

The green checks indicate that the project is enabled, fulfilled, and free of conflicts. If one or more red Xes are displayed in the Node Status tab for your own project, you need to explore the Node list view to determine which children needed to be fulfilled to meet your specific requirements.

You can continue enabling nodes in this sample project in a similar manner until you fulfill all necessary requirements and eliminate unacceptable conflicts.



## APPENDIX **B** Files and Directories

---

This chapter discusses additional details that may be useful to some expert users, although none of the information explained here is essential to using Embedix<sup>®</sup> Target Wizard. Topics included are

- “Project Files and Directories” on page 75
- “Target Wizard Defaults Files” on page 78
- “Linux System Subdirectories” on page 80
- “Package Groups in Target Wizard” on page 82

### Project Files and Directories

For every project you create in Embedix Target Wizard, the system creates two project directories—the actual directory and a symlink directory.

The actual directory:

```
/opt/Embedix/home/<user>/project/<project>
```

The symlink directory:

```
/home/<user>/project/<project>.
```

It then creates in the root project directory the files `ProjectName.epj` and `ProjectName.ess`, where `ProjectName` is the name given to the project. These two files are the basis of any project.

The file with the extension `.epj` is the project settings file, which includes path definitions and other project settings.

The other file is the save-state file, which contains the configuration state (enabled/disabled, integer values, strings, etc.) for every component and option in the project.

A typical project directory has a structure similar to the following:

```
|-- Packages
|   |-- generic
|   |-- local
|-- build
|   |-- dev_image
|   |-- packagestate
|   |-- projectstate
|   |-- rpmdir
|   |   |-- BUILD
|   |   |-- RPMS
|   |   |-- SOURCES
|   |   |-- SPECS
|   |   |-- SRPMS
|   |-- tarfiles
|-- config-data
|   |-- buildcontrol
|   |   |-- generic
|   |   |-- i386_default
|   |   |-- local
|   |-- ecds
|   |   |-- generic
|   |   |-- i386_default
|   |   |-- local
|   |-- specpatches
|-- emb-bin
|-- merge
|-- src
|-- tmp
```

## Packages

There are two directories under Packages. Source RPMs are expected to be located here. *Packages/generic* is for RPMs from the Embedix Linux distribution, and *Packages/local* is for custom packages that are not part of the Embedix distribution proper.



## build

Subdirectory	Purpose
build/dev_image	Once a project is done building, the image for it resides here.
build/packagestate	This directory contains build instructions for each package.
build/projectstate	When saving a project with Target Wizard, the .ess files are added to this directory.
build/rpmdir	All rpm related activity takes place under this directory.
build/tarfiles	Pre-build packages (in the form of tar.gz files) reside here.

## config-data

Subdirectory	Purpose
config-data/buildcontrol	The build-control files (with a .lbc extension) reside under this directory
config-data/ecds	Embedix Component Description files (with a .ecd extension) reside under this directory in the generic and local subdirectories.
config-data/specpatches	Specpatches are patches that are applied to an RPM's .spec file to facilitate Target Wizard's fine-grained, package-building features. These patches reside here.

## merge

This is the directory where one places files that are to be merged with the target image.

## Target Wizard Defaults Files

Target Wizard uses two defaults files:

- ▶ /opt/Embedix/etc/twdefaults
- ▶ ~/.twconfig

The first of these files, /opt/Embedix/etc/twdefaults, contains the following five settings:

Directory	Description
EMBEDIX_DIR	Root of the Embedix directory tree.
PROJECT_DIR	Default project directory.
BUILD_DIR	Directory for building packages.
TARGET_IMAGE_DIR	Destination directory of the build; holds the target root file system.
USER_MERGE_DIR	Merge directory.

These values are used as defaults only when projects are created. Normally you should not need to edit this file.

The second file, ~/.twconfig, resides in your home directory. This file contains settings saved from individual Target Wizard sessions, such as the size of the window and the list of recently saved projects. These settings are fairly self-explanatory.

The .twconfig file is not included in the installation, but will be created the first time Target Wizard is run. You may never need to edit this file because the default settings meet the needs of most users. Also, some of these are set by using dialogs in TargetWizard.

If you use a Windows host development machine, you may find the following information helpful in the event you need to hand-edit this file.

**Table B-1.** Windows Host Machine: Options in the .twconfig file

Option	Description
BUILDER_BROADCAST_PORT=	This is the network port used to dynamically find the virtual machine. By default, the value is 49160. The only reason to assign a new port is if there are other services on your network which use the default value. If you set a new value, you must also put the same port in the file "/opt/Embedix/etc/twdefaults" inside the virtual machine. The format of the line is the same in the Linux file as the Windows file.
BUILDER_DEFAULT_EMBEDIX_DRIVE=	If not set, the default is "Q". Set this if you are already using the Q: drive. The default is no value.
BUILDER_DEFAULT_PROJECT_DRIVE=	If not set, the default is "P". Set this if you are already using the P: drive. The default is no value.
LEAVE_EMBEDIX_DIR_MOUNTED_ON_EXIT=	If set to "YES" the drive pointing to the Embedix directory in the VM will remain after Target Wizard is closed. The default is no value.
LEAVE_PROJECT_DIR_MOUNTED_ON_EXIT=	If set to "YES" the drive pointing to the Project directory in the VM will remain after Target Wizard is closed. The default is no value.
BUILDER1=	You can specify a Linux build VM, which will show up in the "Build Servers" dialog. You can use the "Add" button in the dialog, or enter a description directly in the .twconfig (when TW is not running). List the name, address, port and (optionally) a username.  Example:  BUILDER1=name=db1;ip=208.187.28.50;port=49160;user=bsmith BUILDER2=name=anotherBld;ip=208.187.28.51;port=49160;user=bsmith
BUILDER_DEFAULT=	Determines which builder will be automatically selected in the Builders dialog, and overrides the selection of the local VM ware builder. This can be set by the "Set as Default" button when connecting to a builder. If this is not set, the local builder, if found, will be selected.

**Table B-1.** Windows Host Machine: Options in the .twconfig file

Option	Description
ALWAYS_PROMPT_FOR_LOGIN=	Added for development uses only. By default the user "vkit" is logged-in to the VM. If this option is set to "YES" a username and password dialog will be displayed when connecting to a builder.

## Linux System Subdirectories

The Embedix system directory, the root Linux directory where Embedix tools are located, is normally /opt/Embedix. Under this directory are the following subdirectories:

### **/opt/Embedix/embedix-2.0/Packages**

This is where SRPMs for each package reside.

### **/opt/Embedix/embedix-2.0/config-data/ecds**

This is where ECD files for each package reside.

### **/opt/Embedix/embedix-2.0/config-data/build-control**

This directory contains .lbc files that contain generic package building information. The following headings are allowed:

Heading	Description
%pkg_file	source name (e.g., linux-2.4.2-1.src.rpm) -
%required	
%patches	patches for SRPM and TAR files
%bld_dir_name	build directory name - usually found in SRPM
%info	
%bin	kernel build directory

Heading	Description
%bld_targ	used by kernel only (ie - zImage, vimage, %bzImage)
%makec	instructions for package make configure
%makei	instructions for package make install
%makeb	instructions for make build
%makedc	instructions for make disk clean
%cflags	appended to compile flags
%cfgopts	extra options for configuration stage - glibc
%only	
%spec	spec file name - glibc only

---

### **/opt/Embedix/embedix-2.0/config-data/patches**

This is where specfile patches (for the few packages that need them) reside.

### **/opt/Embedix/embedix-2.0/config-data/preconfigs**

Configuration profiles (files with the .sdf extension) for boards and systems reside here.

### **/opt/Embedix/tools/**

This is where the tool-chain (cross-compilers and other auxiliary development utilities) for each supported architecture resides.

### **/opt/Embedix/emb-bin**

Executables reside here.

## Package Groups in Target Wizard

Every time you create a new project in Embedix Target Wizard, your project's default tree view ("Show Groups" enabled) displays all of the available packages (or components) for your target platform by package group.



**Note:** You can view all components in an alphabetical list in the tree view by disabling the Show Groups option. To enable or disable the Show Groups, choose the menu item View > Show Groups.

---

Click on the arrow next to a group to view the group's components. Click on the arrow next to a group component to view the group component's options and/or components.

Table 1 lists all components (or packages) available in an x86 target platform project by group. For the list of packages available in your board support package (BSP), see the documentation that shipped with the BSP.

**Table 1.** Packages by Target Wizard Group

Target Wizard Group	Subgroup	Full Path to Available Packages
Kernel	(none)	[/Embedix]-kernel
Textprocessing	Editor	[/Embedix]-[Textprocessing]-[Editor]-Include /usr/bin/nano? [/Embedix]-[Textprocessing]-[Editor]-elvis-tiny
Textprocessing	Tools	[/Embedix]-[Textprocessing]-[Tools]-textutils
Server	Server Network	[/Embedix]-[Server]-[Server_Network]-wu-ftpd? [/Embedix]-[Server]-[Server_Network]-Include /usr/sbin/in.identd? [/Embedix]-[Server]-[Server_Network]-nfs server
Server	Server Boot	[/Embedix]-[Server]-[Server_Boot]-Include /usr/sbin/dhcpd?

**Table 1.** Packages by Target Wizard Group

Target Wizard Group	Subgroup	Full Path to Available Packages
Server	WWW	[/Embedix]-[Server]-[WWW]-tthttpd [/Embedix]-[Server]-[WWW]-Include the boa webserver?
Administration	Hardware	[/Embedix]-[Administration]-[Hardware]-pciutils [/Embedix]-[Administration]-[Hardware]-Include /sbin/hdparm?
Administration	Administration Network	[/Embedix]-[Administration]-[Administration_Network]-Include /usr/sbin/tcpdump? [/Embedix]-[Administration]-[Administration_Network]-tcp_wrappers [/Embedix]-[Administration]-[Administration_Network]-Include ipchains binary and init script?
Applications	Communications	(none)
Applications	User Interfaces	(none)
X11	(none)	[/Embedix]-[X11]-Microwindows (Nano-X)
Scientific	Math	(none)
Communication	Communication Network	[/Embedix]-[Communication]-[Communication_Network]-Include /usr/bin/rsync?
Real-Time	(none)	[/Embedix]-[Real-Time]-RTAI
Programming	Programming Tools	[/Embedix]-[Programming]-[Programming_Tools]-Include Flex?
Programming	Interpreters	[/Embedix]-[Programming]-[Interpreters]-perl

**Table 1.** Packages by Target Wizard Group

Target Wizard Group	Subgroup	Full Path to Available Packages
Programming	Debugger	[/Embedix]-[Programming]-[Debugger]-Include r2d2 (zrttb) tools? [/Embedix]-[Programming]-[Debugger]-startkgdb [/Embedix]-[Programming]-[Debugger]-gdbserver [/Embedix]-[Programming]-[Debugger]-Include /usr/sbin/TraceDaemon?
System	Shell	[/Embedix]-[System]-[Shell]-Include /bin/bash? [/Embedix]-[System]-[Shell]-Include /bin/ash?
System	Time	[/Embedix]-[System]-[Time]-Include /usr/share/zoneinfo? [/Embedix]-[System]-[Time]-vixie-cron [/Embedix]-[System]-[Time]-Include /usr/bin/time? [/Embedix]-[System]-[Time]-Include the sample crontab database? [/Embedix]-[System]-[Time]-Include /usr/sbin/atd?
System	Utilities	[/Embedix]-[System]-[Utilities]-tinylogin [/Embedix]-[System]-[Utilities]-Include /sbin/mgetty? [/Embedix]-[System]-[Utilities]-less? [/Embedix]-[System]-[Utilities]-default_passwd [/Embedix]-[System]-[Utilities]-Include cgetty? [/Embedix]-[System]-[Utilities]-busybox
System	Compression	[/Embedix]-[System]-[Compression]-component_gzip [/Embedix]-[System]-[Compression]-bzip2
System	File System	[/Embedix]-[System]-[FileSystem]-findutils [/Embedix]-[System]-[FileSystem]-Include /usr/bin/file? [/Embedix]-[System]-[FileSystem]-ext2fs



**Table 1.** Packages by Target Wizard Group

Target Wizard Group	Subgroup	Full Path to Available Packages
System	Network	[/Embedix]-[System]-[Network]-portmap [/Embedix]-[System]-[Network]-netkit-telnet [/Embedix]-[System]-[Network]-netkit-ftp [/Embedix]-[System]-[Network]-netkit-base [/Embedix]-[System]-[Network]-net-tools [/Embedix]-[System]-[Network]-Include /usr/sbin/micro_inetd? [/Embedix]-[System]-[Network]-iproute2
System	Kernel utilities	[/Embedix]-[System]-[Kernel utilities]-modutils
System	System Tools	[/Embedix]-[System]-[System_Tools]-Include /usr/sbin/lsof? [/Embedix]-[System]-[System_Tools]-gpm? [/Embedix]-[System]-[System_Tools]-diffutils [/Embedix]-[System]-[System_Tools]-Include /usr/bin/strace?
System	Process	[/Embedix]-[System]-[Process]-procps

**Table 1.** Packages by Target Wizard Group

Target Wizard Group	Subgroup	Full Path to Available Packages
System	Library	[/Embedix]-[System]-[Library]-Include tiny terminfo database? [/Embedix]-[System]-[Library]-Include libslang? [/Embedix]-[System]-[Library]-readline [/Embedix]-[System]-[Library]-Include poprt development library? [/Embedix]-[System]-[Library]-ncurses [/Embedix]-[System]-[Library]-Include /usr/lib/libz.so.1.1.3? [/Embedix]-[System]-[Library]-Include libpwdb library? [/Embedix]-[System]-[Library]-Include libpam library? [/Embedix]-[System]-[Library]-Include libgdbm.so? [/Embedix]-[System]-[Library]-Include freetype? [/Embedix]-[System]-[Library]-Include libcrack library? [/Embedix]-[System]-[Library]-glibc
System	Base	[/Embedix]-[System]-[Base]-which? [/Embedix]-[System]-[Base]-util-linux [/Embedix]-[System]-[Base]-Include /etc/termcap? [/Embedix]-[System]-[Base]-tar [/Embedix]-[System]-[Base]-Include system loggers? [/Embedix]-[System]-[Base]-skeleton system directory tree [/Embedix]-[System]-[Base]-sh-utils [/Embedix]-[System]-[Base]-Include /sbin/setserial? [/Embedix]-[System]-[Base]-sed [/Embedix]-[System]-[Base]-Include /usr/bin/passwd? [/Embedix]-[System]-[Base]-Grep [/Embedix]-[System]-[Base]-fileutils [/Embedix]-[System]-[Base]-Device Files

## APPENDIX **C** Windows Build Server Information

---

If you use a Windows host development machine, you may find the following sections useful.

### Finding a Build Server

Target Wizard provides you with the option to dynamically or manually specify a build server. There is also an option to put builder settings in the .twconfig file so the settings do not have to be typed in.

Here is an example of how to specify builders in the .twconfig file:

```
BUILDER1=name=bld1;ip=1.2.3.4;port=10000
BUILDER2=name=bld2;ip=4.3.2.1;port=49160
```

Also, when the "Find" button is hit, by default twnt looks for builders listening on port 49160. To override this value, add the line

```
BUILDER_BROADCAST_PORT=49160
```

to the following files:

```
Windows ~/.twconfig
Linux /etc/twdefaults.
```

If the product is only used with VMware (and not on a dedicated build server), these options may be less important.

### Adding a Build Server Manually

(Coming soon???)



# Index

---

## Symbols

7

.epj suffix 12

## A

additional resources vii

## B

blank display 56

build a project 37

build directory 38

Build Log info area 26

Build menu 21

build options 10, 39, 58

build with conflicts 10, 58

building a target image 38

## C

components

    full path to 82

configuration profile, custom 16

configuration profiles 17

    IA32\_Embedix\_1.0 63

    IA32\_Embedix\_min 63

configurations required 64

conflicting nodes 71

Conflicts info area 26

continue building when errors occur 10, 58

conventions used in manual v

copying files to the user directory 34

create a state file 16

custom application

    merging 34

customizing a project 29

## D

default directory 9, 38, 57

Dependency Conditions list 66

deploying to a board 39, 42

deployment options

    Make a bootable install CD 48

    Make a directory tree 44

    Make a filesystem to chroot into 45

    Make a self-hosting RAM disk filesystem  
    44

    Make install disks for a normal  
    filesystem 47

    Make install disks for a RAM disk  
    system 45

deployment options, diskette 51

deployment options, x86 42

deployment wizard

    starting 40

Description info area 26

directories 57, 80

- directories, project 75
- disabled packages 61
- documentation conventions v

## E

- editing a project 29
- Embedix Target Wizard
  - default files 78
- enabled and fulfilled 72
- epj suffix 12
- etc/twdefaults 78

## F

- File menu 20
- Files info area 26
- files, project 75
- fulfilling node 66

## G

- Generic x86 board target platform 17
- global states 17
- groups 82

## I

- IA32\_Embedix\_1.0 configuration profile 63
- IA32\_Embedix\_min configuration profile 63
- icons 22
- incremental builds 38
- info areas 24

## K

- kernel image 38
- kernel rebuilds 38

## L

- Lipo 31
- loading a configuration profile 64
- loading a preconfigured state 17

## M

- menu bar 19
- merging custom application into target image 34
- merging files into target image 34
- Messages info area 26

## N

- naming a project 10, 57
- navigation tips 27
- new project 56, 60
- Node List window 23
- notes v

## O

- opt/Embedix directory 80

## P

- package groups 82
- packages
  - full paths to 82
- preconfigured state, load 17
- project
  - build 37
  - directories 75
  - edit 29
  - files 75
  - new 56
  - save 30
  - saved states 15
  - size 31

- size reduction 31
- project directory 9, 38, 57
- Project menu 20
- project name 9, 10, 57
- project window
  - columns 24
  - empty 56
  - icons 23
- project, removing 13

## R

- rebuild a project 37
- rebuild all 37, 38
- reducing libraries 31
- related books vii
- required configurations 64
- resolving conflicts 71
- resolving dependencies 69
- root directory 80
- RPM files 37

## S

- sample project
  - part 1 56
- save a project 30
- save state 16
- SDF files 16
- SDFs 15
- Search menu 20
- setting project directories 9
- shared libraries 31
- Size info area 26
- size of project 31
- starting a new project 56
- state definition files 15
- state files 17
- states

- IA32\_Embedix\_1.0 63
- IA32\_Embedix\_min 63
- status bar 23
- system subdirectories 80

## T

- tabs
  - Build Log 26
  - Conflicts 26
  - Description 26
  - Files 26
  - Messages 26
  - Size 26
- target deployment diskette 51
- target image
  - build 38
  - merge 34
- target platforms 17
- Target Wizard
  - default files 78
  - exploring the tree 65
- tips v
- tips on navigating 27
- tool bar 22
- tree view 23

## U

- Update menu 21

## V

- View menu 20
- View option 82
- viewing window
  - populated 60
- virtual machine
  - suspending 7

## W

warnings v  
Web sites

[www.lineo.com](http://www.lineo.com) vii  
[www.lineo.com/services/support/  
index.html](http://www.lineo.com/services/support/index.html) vii