

Creating and managing projects

The purpose of this chapter is to help you understand what an Ebony project comprises and how to manage the projects you create with Ebony. The topics covered in this chapter include

- Project Management
- Using the Object Repository
- Setting project options
- Using the Project Manager
- Navigating among project components
- Managing multiple project versions and team development
- Managing multiple project versions and team development
- Inside the project file

What is project management?

As your applications grow in size and complexity, you'll discover that your programs become more and more dependent on different intermediate files. In addition, you'll find that some of the modules in your project must be compiled with different compilers or different compiler options. A Windows program, for example, can be composed of resource scripts, import libraries, and source code, with each different file type requiring a different compiler and compiler settings.

As the complexity of your projects increase, the need increases for a way to manage the different components or your projects. By studying the files that make up a project, you can see how a project combines different source files to produce different target files. Target files, for example, can be .DLL or .EXE files. The source files that these targets are dependent upon can consist of files like .C, .CPP, and .H files. Project management is the organization and management of the sources and targets that make up your project.

What is a project?

The Ebony application development process centers around the concept of *projects*. An Ebony project is a collection of all the files that together make up a Ebony application (or dynamic-link library). These files are generated as you work on a project at design time.

Project directories

You should store each project in its own directory. Projects can share forms, files, and resources located in almost any directory, but it's best to keep the central project file and any other files specific to the

project in a dedicated directory. See ¹ “Using the Object Repository” on page 101 for more information about project templates and shared forms.

What’s in a project?

To manage projects effectively, you need to understand the various files and file types that constitute a C++ project. In most cases, you don’t need to deal with the individual files. Although you can edit most of these files directly outside of the IDE as source code, it’s easier and more reliable to use the visual editors and other tools in the Ebony integrated development environment.

Project file

Every Ebony project contains C++ source code that Ebony compiles into the finished application or dynamic-link library. The central point for the project’s source code is called the project file.

A very simple project could contain nothing besides the project file, but more often the project file contains references to all the forms and units used by a more complex project. When you load, save, or compile a project, Ebony knows which other files to act on by looking at the project file.

Ebony project files have the extension .CPP and must be valid C++ code. Ebony uses project files which always have a matching makefile which has the same name but with a .POF extension. The makefile stores the various project options and can be used to build the project at the command line.

When you compile or run the project, Ebony’s compiler produces either an executable file or a dynamic-link library on disk with the same name as the project file, but with the extension .EXE or .DLL, as appropriate.

Forms

Forms are a very visible part of most Ebony projects. Normally, you design forms with the Forms Designer in the Ebony integrated development environment, which stores a binary description of the designed forms in form files.

You can open a form file in the Code editor and modify a text version of the same data, or you can translate your form files to text versions for easier maintenance and version control. You can also switch back and forth between the form and text views of a form using the View As Text and View As Form options on the local menus of the form or the Code editor, respectively. Outside of the IDE, you can use CONVERT.EXE” to do similar conversions.

Each form in an Ebony project also has an associated unit. The unit contains the source code (a pair of .CPP and .H files) for any event handlers attached to the events of the form or the components it contains. A unit associated with a form is sometimes called a form unit. Please see the following section on Units for further information.

When you first save a form unit or a project containing unsaved forms, Ebony prompts you to enter a name for each unit, which it uses as the name of the unit file, appending the extension .CPP. The form

¹ x-ref page change “Using the Object Repository.

file gets the same name, but with the extension .DFM. You can use C and C++ (.CPP) extensions for your unit files, but Ebony expects the .DFM extension on the corresponding form file.

Units

Ebony's C++ language supports separately compiled modules of code called units. Using units promotes structured, reusable code across projects. The most common units in Ebony projects are form units, which contain the event handlers and other code for the forms used in Ebony projects. But units don't have to have forms associated with them. They can contain any kind of C or C++ code you want to write.²

When you compile a unit, Ebony's compiler produces an object file with the same names as the source file but with a .OBJ extension. You should never need to open these binary files, and you do not need to distribute them with the completed project.

³Units are also used when you create new components, as described in the⁴ Ebony Component Writer's Guide.

Resource files

Ebony uses standard Windows-format resource files to include items such as the application icon in a project. You can edit these resource files (or create your own) to add other resources, such as bitmaps, cursors, icons, or strings to the application.

By default, each Ebony project that compiles into an application has a resource file with the same name as the project file, but with the extension .RES.

Other items

Ebony generates other files in conjunction with maintaining and compiling your project, most of which you never need to consider. These include:

- Project options file

The project options file is a text file containing the current settings for project options, such as compiler and linker settings, directories, conditional directives, and command-line parameters. In most cases, you'll set these options using the Project Options dialog box, but Ebony saves them in text form for easy maintenance, version control, and sharing.

The project-options file has the same name as the project file, but with the extension .POF.

- Desktop settings

This text file stores information about the state of your project, such as which windows are open, and in what positions, so you can restore your workspace on a project-by-project basis.

The desktop-settings file has the same name as the project file, but with the extension .DSK.

² The example in Chapter 14 shows the use of a code-only unit to include some commonly used routines.

³ See Chapter 6, "Writing C ++ Code" for more information about units.

⁴ x-ref change name to Ebony Component Writers Guide (Is this book in the sku for Ebony?)

Naming unit and project source code files

As you open new units in a project, Ebony supplies them with default names such as UNIT1.CPP, UNIT2.CPP, UNIT3.CPP, and so on. You will soon run into confusion if you use these default names when saving the project. It is highly recommended that you supply a meaningful name for each unit in a project as you save it.

When naming the project source code file, you need to supply a name not previously used by any of the unit files already in the project file. If you try to give the project the same name as one of these units, Ebony displays an error message. If this occurs, respond NO to the error message box and supply a different name for the unit in the Save As dialog box.

As with unit files, Ebony supplies a default name, PROJECT1.CPP, for the project file. You should supply a more meaningful name the first time you save the project.

All unit and project file names must be legal C++ identifiers. When the compiler looks for a unit or project file, it first searches for a file with the full name of the unit or project identifier. If it does not find that file, it will then search for a version of the identifier name, truncated to eight characters. This is for backward compatibility and for compatibility with file servers that only store short file names. You should not manually truncate your file names.
projects:overviewproject filesfile types
(project)project files:types

About project options

There are a number of ways in which you can customize the way Ebony appears and works for a particular project, including customizing the integrated development environment (IDE) itself. The appearance and behavior of the IDE when you start Ebony or begin a new project are governed by the settings of several types of options:

- Environment settings affect all Ebony projects.
You set those in the Environment Options dialog box (Options|Environment).
- Repository settings determine the default new form and new project.
Set those in the Object Repository Options dialog box (Options|Repository).
- Project settings affect the current project only.
Set those in the Project Options dialog box (Options|Project).

Note

If you share your installation of Ebony with other users, it's possible that another user has modified the default option settings; in that case, displays or behaviors might differ from those described in the examples and illustrations in this chapter. In a shared-installation situation, it's a good idea to check and modify environment options as described in the following sections, before creating a new project.

Using the Object Repository

Ebony's Object Repository provides a versatile mechanism for sharing forms, dialog boxes, and data modules across projects. It can also help with reusing similar forms in a single project, and provides project templates as starting points for new projects.

This section focuses on how to use the Object Repository in general as a project management tools and discusses some of the mechanics of using project templates.⁵ ⁶In addition to some general guidelines on using the Object Repository, this section discusses

- Object Repository usage options
- Using project templates
- Customizing the Object Repository

Changing defaults for new projects

About the Object Repository

Ebony provides the Object Repository as a means for sharing and reusing forms and projects. The repository itself is really just a text file that contains references to forms, projects, and experts. Details of the file format are in online Help.

Sharing across projects

By adding forms, dialog boxes, and data modules to the Object Repository, you make them available to other projects. For example, in a simple case, you could have all your projects use the same About box, copied from the Object Repository. A more advanced use of the Object Repository would be to have a standard empty dialog box with the company or product logo and standard button placement, from which all your projects derive standard-looking dialog boxes.

These sharing options are described in detail in “Object Repository usage options.”

Sharing within projects

The Object Repository can also help you to share items within a project, by allowing you to inherit from forms already in the project. When you open the New Items dialog box (by choosing File|New), you’ll see a page tab with the name of your project. If you click that page tab, you’ll see all the forms, dialog boxes, and data modules in your project. You can then derive a new item from the existing item, and customize it as needed.

For example, in a database application you might need several forms that display the same data, but which provide different command buttons. Instead of creating and maintaining several nearly-identical forms, you could lay out a generic form that contains all the data-display controls, then create separate forms that inherit the data-display layout, but have different command buttons.

By carefully planning your project forms, you can save tremendous amounts of time and effort by sharing forms within projects.

⁵ x-ref For details on using the Object Repository for forms and dialog boxes (what chapter was Chapt. 2)

⁶ x-ref For information on using the Object Repository with data modules, see the DatabaseApplication Developers Guide.

Sharing entire projects

You can also add an entire project to the Object Repository as a template for future projects. If you have a number of similar applications, for example, you can base them all on a single, standardized model.

Using experts

The Object Repository also contains references to experts, which are small applications that lead the user through a series of dialog boxes to create a form or project. Ebony provides a number of experts, and you can also add your own.

Object Repository usage options

When you use an item from the Object Repository in a project you have as many as three options on how to include that item. Keep in mind that items in the Object Repository are there to be shared, and that you want to use them in ways that help, rather than hinder, reuse.

In general, you these are the three options for using Object Repository items:

- Copy the item
- Inherit from the item
- Use the item directly

Copying items from the Object Repository

The simplest sharing option is to copy an item from the Object Repository into your project. Copying makes an exact duplicate of the item as it stands and adds the copy to your project. Future changes to the item in the Object Repository will not be reflected in your copy, and alterations made to your copy will not affect the original Object Repository item.

Copying is the only option available for using project templates.

Inherit from Object Repository items

The most flexible and powerful sharing option is to inherit from an item in the Object Repository. Inheriting derives a new class from the item and adds the new class to your project. When you recompile your project, any changes made to the item in the Object Repository will be reflected in your derived class, unless you have changed a particular aspect. Changes made to your derived class do not affect the shared item in the Object Repository.

Inheriting is available as an option for forms, dialog boxes, and data modules, but not for project templates. It is the only option available for reusing items from within the same project.

Using Object Repository items directly

The least flexible sharing option is using an item from the Object Repository directly in your project. Using the item add the item itself to your project, just as if you had created it as part of that project. Design-time changes made to the item therefore appear in all projects that directly use the item, as well as affecting any projects that inherit from the item.

Using items directly is an available option for forms, dialog boxes, and data modules, but you should use it sparingly.

Items shared this way should generally be modified only at run time, to avoid making changes that affect other projects.

Note

The Copy option is the only option available for experts, whether form experts or project experts. Using an expert doesn't actually add shared code, but rather runs a process that generates its own code.

Using project templates

Ebony provides project templates, predesigned projects you can use as starting points for your own projects. Project templates are part of the Object Repository (located in the OBJREPOS subdirectory), which also provides form objects and experts.

When you start a project from a project template (other than the blank project template), Ebony prompts you for a *project directory*, a subdirectory in which to store the new project's files. If you specify a directory that doesn't currently exist, Ebony creates it for you. Ebony copies the template files to the project directory. You can then modify it, adding new forms and units, or use it unmodified, adding only your event-handler code. In any case, your changes affect only the open project. The original project template is unaffected and can be used again.

To start a new project from a project template,

- 1 Choose File|New to display the New Items dialog box.
- 2 Choose the Projects tab.
- 3 Select the project template you want and choose OK.
- 4 In the Select Directory dialog box, specify a directory for the new project's files.

A copy of the project template opens in the specified directory.
projects:building:newcreating:new projectsproject
templatetemplates:projectspredesigned projects

Adding projects to the Object Repository

You can add your own projects and forms to those already available in the Object Repository. This is helpful in situations where you want to enforce a standard framework for programming projects throughout an organization.

For example, suppose you develop custom billing applications. You might have a generic billing application project that contains the forms and features common to all billing systems. Your business centers around adding and modifying features in this application to meet specific client requirements. In such a case, you might want to save the project containing your Generic Billing application as a project template and perhaps specify it as the default new project on your Ebony development system. Likewise, you'll probably have a particular form within this project that you want to appear as the default main or new form.

To add a project to the Object Repository,

- 1 Open the project you want added to the Object Repository.
- 2 Choose Project | Add to Repository to invoke the "Add to Repository" dialog.
- 3 On the Project Templates page, choose Add to display the Save Project Template dialog box.

- 4 In the Title edit box, enter a project title.
The title for the template will appear in the Object Repository window.
- 5 In the Description field, enter text that describes the template.
This text will appear in the Object Repository window's status bar.
- 6 In the Page field, choose the name of the page in the New Items dialog box (probably Projects) you want the template to appear on.
- 7 In the Author field, enter text identifying the author of the application.
Author information appears only when the user views the repository items with full details.
- 8 Choose Browse to select an icon to represent this template in the Object Repository.
- 9 Choose OK to save the current project as a project template.

Note

If you later make changes to a project template, those changes automatically appear in new projects created from that template. They will not, however, affect projects already created from that template.

You can also save your own forms as form templates and add them to those already available in the Object Repository. This is helpful in situations where you want to develop standard forms for an organization's software, as in the earlier example.

To add a form to the Object Repository as a template right click on the form and choose "Add to Repository, then follow the preceding steps for adding projects as templates, using the Form Templates page of the Object Repository Options dialog box instead of the Project Templates page.

Customizing the Object Repository

The settings in the Object Repository Options dialog box affect the behavior of Ebony when you begin a new project or create a new form in an open project. This is where you specify

- Default project
- Default new form
- Default main form

You always have to option to override these defaults by choosing File|New and selecting from the New Items dialog box.

By default, opening a new project displays a blank form. You can change this default behavior by changing Object Repository options.

Specifying the default new project

The default new project opens whenever you choose New Application from the File menu on the Ebony menu bar. If you haven't specified a default project, Ebony creates a blank project with an empty form. You can specify a project template (including a project you have created and saved as a template) as the default new project.

You can also designate a project expert to run by default when you start a new project. A project expert is a program that enables you to build a project based on your responses to a series of dialog boxes.

To specify the default new project,

- 1 Choose Options|Repository to display the Object Repository dialog box.
- 2 Choose Projects in the Pages list.
- 3 Select the project object you want as the default new project from the Objects list.
- 4 With the object you want selected, check New Project.
- 5 Choose OK to register the new default setting.projects:default:specifying

Specifying the default new form

The default new form opens whenever you choose File|New Form or use the Project Manager to add a new form to an open project. If you haven't specified a default form, Ebony uses a blank form. You can specify any form template, including a form you have created and saved as a template, as the default new form. Or you can designate a form expert to run by default when a new form is added to a project.

To specify the default new form for new projects,

- 1 Choose Options|Repository to display the Object Repository dialog box.
- 2 Choose Forms in the Pages list.
- 3 Select the form object you want as the default new form.
- 4 With the object you want selected, check New Form.
- 5 Choose OK to register the new default setting.

Specifying the default main form

Just as you can specify a form template or expert to be used whenever a new form is added to a project, you can also specify a form template or expert that should be used as the default main form whenever you begin a new project.

To specify the default main form for open projects,

- 1 Choose Options|Repository to display the Object Repository dialog box.
- 2 Choose Forms in the Pages list.
- 3 Select the form object you want as the default main form.
- 4 With the object you want selected, check Main Form.
- 5 Choose OK to register the new default setting.forms:specifying defaultObject Repository Options dialog box

Using the Object Repository in a shared environment

To change the location where Ebony looks for the Object Repository File, EBONY.DRO, create a new String Value called "Base dir" in the "HKEY_CURRENT_USER|Software\Borland\Ebony\1.0\Repository" key in the Windows Registry Editor and set its Value data to the directory where this file is to be located. It is recommended that the location uses the UNC name when the EBONY.DRO file is to be shared.

It is suggested that Forms and Projects be saved using UNC names when they will be added to a shared Repository.

While someone is modifying the Object Repository (EBONY.DRO file), anyone attempting to open or add to the repository will get a dialog box with the user name of the person that has the repository open. If you are attempting to open the repository from Tools| Object Repository, the dialog box will ask if you wish to view the repository. If you choose to view the repository, you will not be allowed to save any changes.

Lock information is stored in the EBONY.DRL file. If this lock file cannot be opened, an exception is raised. This can mean the file is read-only or the user doesn't have write rights for the directory. Also, if someone exits from Ebony abnormally while modifying the repository, the lock file may still contain information that the user is editing in the repository and you will not be allowed to modify the EBONY.DRO file. In this case the lock file should be deleted.

Setting project options

The settings in the Project Options dialog box affect only the current open project. An options file with a file extension .POF is saved in the project directory each time you save the project. When you reopen the project in future work sessions, the project options, as saved in the options file, are in effect.

To display the Project Options dialog box, choose any of these methods:

- In the Project Manager, choose the Options button on the toolbar.
- In the Project Manager, Right-click and choose Options.
- Choose Options|Project from the Ebony menu bar.

The next sections point out the options in the Project Options dialog box that pertain to project management. Detailed information for each option in the dialog box can be found in online Help.

Changing defaults for new projects

The Project Options dialog box contains a check box labeled Default. This control enables you to modify some of Ebony's default project configuration properties. Checking this control writes the current settings from the Compiler, Linker, and Directories/Conditionals pages of the Project Options dialog to a file called DEFAULT.POF. Ebony creates this file when you check the Default box and choose OK in the Project Options dialog box. Ebony then uses the project options settings stored in this file as the default for any new projects you create.

If you create a project from a template in the Object Repository that has its own options file, those settings will override the default settings in DEFAULT.POF. DEFAULT.POF is required for making new projects.

Note

Project options you set for an open project override the current Ebony defaults, whether those defaults are as originally shipped or as modified by you or another user.

Form options

The options on the Forms page of the Project Options dialog box enable you to change the form designated as the project's main form, and to control the creation order of forms during application initialization.⁷

Application page

The Application page is where you specify a title, Help file, and icon for the project. You can also specify these options as default project settings. You can also read or set these items in code at run time, using the Title, HelpFile, and Icon properties of the Application object.

Title

The text you enter here appears below the application icon when you minimize the running application. If no title is specified, the minimized application displays the file name of the .EXE file.

Help file

If you create a Windows Help (.HLP) file for the project, you “connect” it to the application by specifying it in the Help File edit box. Either type a file name or use the Browse button to select an existing Help file.

At run time, the Help file name is used by the HelpCommand, HelpContext, and HelpJump methods of the Application object. The application's Help methods all call the Windows API function WinHelp, passing the name of the Help file.

Caution

Be aware that the Help file for your distributed application might not reside in the same location, relative to the executable file, as it does on your development system. Avoid specifying a drive name unless you know for certain that the finished application's Help file will always reside on a drive with that name.

Consult your Windows Help documentation for further information on how WINHELP.EXE attempts to locate Help files.

Icon

Ebony supplies a default icon for all projects, which appears whenever the running application is minimized, or when the application file appears in the Windows Explorer or Program Manager. You'll probably want to use a different icon for your applications. The Ebony Image Library supplies some icon files that you can use directly or modify by using the Ebony Image editor. You can specify any standard Windows icon file residing in any location. The specified icon is compiled into the project's executable file.

⁷ x-reference change to correct Chapter #

Compiler page

The options on this page enable you to specify compiler options. Modified settings affect only the current project unless the Default box is checked. Checking the Default checkbox + OK will cause the current settings for all three pages (compiler, linker, directories/conditionals) to be written to DEFAULT.POF.

For information on the individual settings on this page, see online Help.

Linker page

This page enables you to specify how the link options are configured.

Checking the Default checkbox + OK will cause the current settings for all three pages (compiler, linker, directories/conditionals) to be written to DEFAULT.POF.

Directory and Conditional options

On the Directories/Conditionals page, you can control the location of the project's compiled output and specify the location of resources needed to compile, link, and distribute the application.

If you want the compiled project output to be stored in a directory other than the project directory, you can enter a path in the Output Directory edit box. The specified directory must exist or be accessible at compile time or Ebony generates an error message.

The Search Path edit box specifies the location(s) of any non-Ebony library file(s) needed to compile the program. If your program needs to access libraries (include files) that don't reside in the Ebony "INCLUDE" and "LIB" directories, you need to specify the library path location in this area of the dialog box. See online Help for more information.

Enter any symbols referenced in conditional compiler directives in the Conditional Defines edit box. Again, see online Help for information.
projects:optionsIDE:customizingintegrated
development environment:customizingProject Options dialog box

Using the Project Manager

The Ebony Project Manager provides a high-level view of the form, unit files, resource, object, and library files listed in the project file. You can use the Project Manager to open, add, save, and remove project files, and to open the Project Options dialog box, where you can configure project default settings.

If you share files among different projects, using the Project Manager is recommended because you can quickly and easily tell the location of each file in the project. This is especially helpful to know when creating backups that include all files the project uses.⁸(See page 5-44 for more information on making backups.)

⁸ x-ref change to correct page # reference.

You must have a project open in order to view the Project Manager.
To open the Project Manager window, choose View|Project Manager.

The Project Manager window

The Project Manager window displays information about the status and file content of the currently open project. It also provides quick access to project management functions, easy navigation among the constituent files, and access to project options through the toolbar and context menu.

Figure 5.1 Project Manager window

The main elements of the Project Manager window are:

- The toolbar
- The project file list
- The project status bar

The Project Manager also has a context menu you can display by Right-clicking anywhere inside the window.

Project Manager toolbar

The toolbar has six buttons that provide quick access to common project tasks. Table 5.1 describes each button and its use.

Table 5.1 Project Manager toolbar buttons

Button	context menu command	Ebony menu command	Function	Comment
XE "Add button (Project Manager)"XE "Add File command (Project Manager)"XE "project files:adding"Add	Add File	File Add To Project	Adds a shared or non-shared file to the project. The Path column of the Project Manager's file list reflects the location of any shared file.	Non-shared files reside in the project directory. Shared files still reside outside the project directory; they are not copied to the current project directory. Any changes to the shared file, in any project, are reflected in <i>all</i> projects using the file.
XE "Remove button (Project Manager)"XE "Remove File command	Remove File	File Remove From Project	Removes the selected file(s) from the current project.	Only the relationship of the selected file(s) with the current project is removed. The file still exists in its current location. The project file is updated to reflect the change.

d (Project Manager)"XE "project files:rem oving from project" Remove				
XE "View Unit button (Project Manager)"XE "View Unit comman d (Project Manager)"View Unit	View Unit	View Units (Ctrl + F12)	Opens and displays the selected file in the Code editor. If multiple files are selected in the Project Manager window, displays the most recently selected file.	Using the Ebony menu command opens the View Unit dialog box, where you can select which unit to open.
XE "View Form button (Project Manager)"XE "View Form comman d (Project Manager)"View Form	View Form	View Forms (Shift + F12)	Displays the visual image of the currently selected form unit. If the selected file is not a form unit, this button and the context menu command are disabled.	Using the Ebony menu command opens the View Form dialog box, where you can select which form to display.
XE "Options button (Project Manager)"XE "Options comman d (Project Manager)"Option s	Options	Project Options	Displays the Project Options dialog box.	
XE "Update	Update		Synchronizes the Project Manager window display with listings in	This button normally remains disabled, and its parallel context

button (Project Manager)"XE "Update comman d (Project Manager)"Update	the project source code file.	menu command dimmed, unless you have manually modified the project file (which is not recommended).
--	-------------------------------	--

Project Manager file list

This area of the Project Manager details the current composition of the project. It gets its information from the project file. If you have modified this file manually (that is, you edited the source code directly), the information in this list might be inaccurate. When you open the Project Manager, Ebony compares the information in the project file to the last saved information for the Project Manager. If these are not synchronized, the contents of the Project Manager file list become dimmed, and the Update button on the toolbar becomes enabled so that you can synchronize the information.

Caution

Ebony has mechanisms for automatically tracking the files that make up a project and for keeping the project file updated. Avoid editing project files manually unless you have a thorough understanding of this process and its ramifications. By editing a project file, you circumvent Ebony's automated project management and risk maintaining inaccurate information about project components. Compilation failures and other problems can result.

Using the Project Manager is highly recommended because it enables you to track the location of everything in your project.

Project Manager status bar

This area at the bottom of the Project Manager window displays the full path name of the project file, and indicates the number of forms and units in the project.

The project file path name can be a useful reference if you are bringing many forms and units that reside in locations other than the main project directory into the current project. The form and unit summary information can be helpful in evaluating the scope of proposed project modifications. Project Manager

Integrating forms and units into a project

You can use either the Project Manager or commands on the Ebony File menu to create new forms or new unit files in a project, or to add existing files from locations outside the project directory. For any form or unit to "belong" to a project, the project must be open so that Ebony can update the project file as new or shared unit files are added.

As you build a project, you can

- Create new forms
- Create new source code units
- Use files from a different project or location (shared file

- Use existing Borland C++ and Pascal units

This section discusses each of these topics.

Creating new forms

Opening a new form in a project is one of the tasks you probably do most frequently.⁹

Creating new source code units

If you write custom C++ procedures and functions as a stand-alone routines callable from any unit, you need to save them in new source code units. If you want to create this type of unit and not have it be part of a project, you can open the new unit without having a project open.

To open a new source code unit,

- 1 Choose File|New from the Ebony menu bar.
- 2 Select Unit in the New Items dialog box.
- 3 Choose OK.

If you have a project open, you can also choose New Unit from the Project Manager context menu. When you create a unit in an open project with the New Unit command, the new unit is registered in the project file.

Sharing files from other projects or directories

A project can use existing form and unit files that it did not create and which don't reside in the current project directory, or any subdirectory of that directory. When you compile your project, it does not matter whether the files that make up the project reside in the project directory, a subdirectory of the project directory, or any other location.

If you add a shared file to a project, bear in mind that the file is not copied to the current project directory; it remains in its current location. Adding the shared file to the current project simply registers the file name and path in the project file. Ebony automatically does this as you add units to the project. The compiler treats shared files the same as those created by the project itself.

To add a shared file to the current project, do one of the following:

- Choose the Add File button on the Ebony toolbar.
- Choose the Add button on the Project Manager toolbar.
- Choose Add File from the Project Manager context menu.

Any of these actions displays the Add To Project dialog box, in which you can select the file you want the current project to use. The Path column of the Project Manager's file list displays the path to the shared file. `files:sharingunit files:sharingproject files:sharingsharing project files`

Using Borland C++, C or Pascal source code units

If you have existing source code units for custom procedures or functions written in Borland C++, C or Pascal you can use these units in an Ebony project. You add these files in the same way as files created in Ebony. `building projects:building`

⁹ x-ref change to correct Chapter #

Removing files from a project

You can remove forms and units from a project at any point during project development. The removal process deletes the reference to the file from the project file. Using the following procedure to remove a unit that has an associated form also removes the form from the project.

To remove a unit from a project, open the project and choose any of these methods:

- In the Project Manager window,
 - Select the unit or units you want to remove.
 - Choose the Remove button on the toolbar or choose Remove File from the context menu.
- Choose the Remove File From Project button on the Ebony toolbar.
- Choose File|Remove From Project from the Ebony menu bar.

Removing a file from the project ends its relationship with the project; it does not delete the file from disk.

Note

Ebony will not let you remove the project (.CPP) file for a project.

Caution

Do not use Windows file management programs to delete Ebony project files from disk until you have performed the preceding removal process in every Ebony project that uses the files. Otherwise, the project file of each project using the deleted files retains references to them. When you open the project again, Ebony will attempt to find the deleted files and display error messages for each file it cannot find. When the project opens, the information about its constituent files in the Project Manager is inaccurate.

Saving projects and individual project files

At any time during project development, you can save an open project in its current state to the project directory. You can optionally save a copy of the project in a different directory under the same or a different name.

You can also save your project as a project template which adds it to the Object Repository so that you or others can reuse it. For more information, see¹⁰ “Adding projects to the Object Repository” on page 5-32.

You are not limited to saving a project as a whole. Ebony enables you to save individual constituent files of a project, including saving a copy of a file to a different directory or under a different file name.

Saving a project

This section explains how to save an open project to the directory created to store it (that is, the *project* directory).

¹⁰ x-ref change page # reference.

Note

If the project was begun from a project template, the Object Repository selection process creates the project directory. Otherwise, Ebony saves projects by default to the current working directory, unless you specify otherwise.

To save all open project files to the project directory, use one of the following methods:

- Choose File|Save All.
- Choose the Save Project button on the Ebony toolbar.
- Choose the Save Project command on the Project Manager context menu.

From here, the save process for projects varies somewhat depending upon whether you have previously saved the project:

- If you have not previously saved the project, Ebony displays the Save As dialog box. This dialog box prompts you to supply a name for each open unit file that has been created in the current project. (You are not prompted for any shared files you might have added to the project. See “Sharing files from other projects or directories” earlier in this chapter.)

After you name the unit file(s), Ebony prompts you to name the project file before saving it to disk. This processing order ensures that the unit and form file names you just specified are correctly registered in the project file’s source code.

- If you have previously saved the project, all open files that reside in the project directory are saved to disk if they have been modified.

If you have opened any new forms or units in the project since the last save, the Save Unit As dialog box appears and prompts you to name those unit files before saving them.

The project file is then updated to reflect any new units and any newly shared files that you have specified for the project to use.

Saving a copy of the project file

Ebony enables you to save a separate version of an open project in a directory other than the project directory. The File|Save Project As command initiates the process. However, because the open project might use shared files in addition to files that were created as part of the current project, the Save Project As command saves only a copy of the project file, project options settings, and the project resource file to the new location.

Important

No unit files are saved to the new location. When you open the copied version, the Project Manager displays all units in the copied project as shared files; that is, none of them reside in the project directory of the currently open project.

To save a separate copy of the project file in another location,

- 1 Choose File|Save Project As from the Ebony menu bar to display the Save <projectname> As dialog box.
- 2 Select the directory where you want to copy the project file.
- 3 If you want to save the project file under a different name, enter the new name in the File Name edit box. If a project file with the same name exists in the directory you specify, you’re prompted as to whether you want to overwrite the existing file.

4 Choose OK to complete the task.

The open project is now the project you just saved.

Ebony saves the project file, the project options file, and the project resource file under the name and/or new location you specify. Ebony also saves any modified unit files (in their current location), so you won't be prompted to save these changes again when you close the project. When you open either version of the project, all changes saved with the Save As operation are reflected in both places.

If you check the file list in the Project Manager, you will see that all the files in the currently open version of the project reside in a directory other than the current project directory. If you want separate copies of any of those files in the new project directory, you need to save them individually to the new location using File|Save As.¹¹ (See "Creating a backup of an entire project" on page 5-44 for more information.)

If you leave the new project unchanged, it continues to use the files in their present (that is, old) location as shared files, which might or might not be what you want. If you don't understand how the new project is using its files, you can run into problems later.

Caution

Do not use file management tools other than those in Ebony to save a copy of a project to a new location.

Saving individual files

You can save individual files in a project, or non-project files (such as text files) that you might have open in the Code editor. To save a file, it must be open.

To save an individual file,

- 1 Bring the file to the front of the Code editor by selecting its tab.
- 2 Choose File|Save File. If this is the first time you've saved the file, you're prompted to name it.
- 3 If necessary, name the file and choose OK.
Ebony saves the file.

To save a file under a different name or location,

- 1 Bring the file to the topmost level of the Code editor by selecting its tab.
- 2 Choose File|Save File As.
The Save File As dialog box appears.
- 3 Specify the new file name, or location, or both, and choose OK.
Ebony saves a copy of the file under the name and location you specify.

Note

This changes the name of the file, and if it is already part of the project, includes the file with the new name in the project. The older file still exists, but isn't included in the project any longer.**project files:savingprojects:savingsaving:project**

¹¹ x-ref change to correct page # reference

Creating a backup of an entire project

Backing up a project can be a simple matter of copying directories or can involve some additional steps. This depends upon how your project directories are structured and whether the project uses files from outside its own directory tree.

The project directory isn't encoded into the project file. The project file does, however, record the location of all the other files in the project. If these files reside in subdirectories of the main project directory, all path information is relative, which makes backup easy. You could back up such a project by copying the directory tree to another location. If you open the project at the backup location, all the project files that reside within that structure are present, and the project will compile.

If this project uses files that reside outside this tree, the project might or might not compile at the backup location. Check the Project Manager's file list to see if these outside files are accessible from the backup location. If they are, the project will compile. If other backup processes already preserve these outside files, then there is probably no need to make separate backup copies of them in the backup project directory.

Navigating among project components

As you work on a project, you will find that you frequently need to navigate back and forth among forms, units, and the various open windows such as the Project Manager, Alignment palette, and so on. This section explains the basic techniques involved.

You can easily toggle between viewing the currently displayed form and its unit source code using commands from the View menu, a keyboard shortcut, or the Project Manager.

You can also open or edit any open unit or form by choosing commands from the View menu, buttons on the Project Manager toolbar, or commands from the Project Manager context menu.

Toggling between form image and unit source code

To switch between viewing the current form and its unit source code, use any of the following methods:

- Press F12.
- Choose View|Toggle Form/Unit.
- Click the Toggle Form/Unit speed button on the Ebony toolbar.
- In the Project Manager, double-click in the Unit column of the unit you want to display its source code, or the Form column to display the form image.

Bringing a window to the front

If you have a number of windows open, such as the Ebony Project Manager or Object Browser, you can easily get to the window you want by selecting it in the Window List dialog box. This dialog box displays a list of all open windows, and enables you to bring any of them to the front.

To bring a window to the front,

- 1 Press Alt+0 (zero) or choose View|Window List from the menu.

2 Double-click the name of the window you want to bring to the front.

Figure 5.2 The Window List dialog box

Using the Project Manager to view or edit units

To view or edit a specific unit,

- 1 Select the unit you want to view or edit in the Project Manager window.
- 2 Choose the unit with any of the following methods:
 - Press the Enter key.
 - On the toolbar, choose the View Unit button.
 - In the Project Manager Right-click and choose the View Unit command.
 - Double-click the unit file name you want in the Unit column.

If the source code file for the selected unit is not open, a Code editor page opens to display it. Otherwise, the appropriate Code editor page comes to the top.

To view or edit a specific form,

- 1 Select the form you want to view or edit in the Project Manager window.
- 2 Choose the form with any of the following methods:
 - On the toolbar, choose the View Form button
 - Press Shift+Enter.
 - In the Project Manager Right-click and choose the View Form command.
 - Double-click the form name in the Form column.

If the form is already open, it comes to the front in design mode. Otherwise, the form opens and comes to the front.

Managing multiple project versions and team development

When you are developing a complex programming project in a team setting, or managing several development projects, you might soon develop the need for a version control system (VCS). A version control system can archive files, control access to project files, and track multiple versions of your projects.

Ebony Developer has a built-in interface for Intersolv's PVCS Version Manager (available separately). It supports PVCS version 5.1 and later. Because Ebony uses an Open Tools API, it can also be used with any commercially available version-control utility. Ebony Client/Server Suite includes both the PVCS interface and PVCS itself.

Enabling team development support

Before using Ebony's built-in team development support, you need to have PVCS installed on your system, and the several files must reside in your \WINDOWS\SYSTEM directory.

For information on using the team development support features of Ebony, look up PVCS or Version Control in Ebony's online Help. You can also read the online Help for PVCS, in the file

PVCS.HLP.project files:sharing sharing project files applications:version control version control

Inside the project file

The project file is the main program file Ebony uses to compile and link all other units and files. Ebony updates this file throughout the development of the project.

Important

Because Ebony maintains the project file, you should not normally need to modify it manually, and in general it is not recommended that you do so. Most changes you could make to the project file can also be made with the Project Manager, and doing so ensures that Ebony keeps all the project's files synchronized.

When you initially save the project, Ebony prompts you to name the project source code file (after being prompted to name any unit source code files the project contains). Since the project file is stored by the operating system, the file name must conform to Windows file-naming rules.

Viewing the project file

As with unit source code files, you can open the project file in the Code editor. The main reason to do this is so you can see the units and forms that make up the project, and which form is specified as the application's main form. As you add forms and units to the project, you can see Ebony's updates of the project source code.

To open the project file, use either of these methods:

- Right-click in the Project Manager and choose the View Project Source command.
- From the Ebony menu bar, choose View|Project Source.

The contents of the project file appear in a page in the Code editor. To hide the project file again, close the page in the Code editor.

Ebony generates the following source code for a default, blank project:

```
#include <vcl.h>
#pragma hdrstop

USEFORM ("Unit1.cpp", Form1);

WINAPI WinMain (HINSTANCE, HINSTANCE, LPSTR, int )
{
    Application->Initialize( );
    Application->CreateForm(_classid(TForm1), &Form1);
    Application->Run( );

    return 0;
}
```

The default project code performs the following:

- The vcl.h header file contains definitions that are needed for the Visual Component Library. This file must be included in each Ebony project.
- The #pragma hrdstop statement instructs the compiler to stop adding header files to the collection of precompiled headers. By default, vcl.h is included in the precompiled headers.
- The WinMain call contains the main source-code block for the project.

Application->Initialize(); initializes the VCL Application object for this process.

The Applicatio->CreateForm statement createss the form specified in the funcion parameters . The statement creates the form object specified in the second parameter using the form class specified in the first parameter.

Ebony adds an Application->CreateForm statement to the project file for each form you add to the project. The statements are listed in the order the forms are added to the project. This is the order that the forms will be created in memory at run time. If you want to change this order, do not edit the project source code. Use the Project|Options menu command.¹² The Application->Run(); statement causes the process to enter the program's message loop. From there, the the process gets Windows messages and dispatches them to the proper window message procedure. This is essentially where your program begins running in the Windows environment.

.DPR filesproject filesSummary

This chapter discussed a number of aspects of project management, including defining projects and their components, and project files. It also covered the various compiler options: compiling, building, and syntax-checking. The chapter also discussed how to use some of Ebony's tools to manage projects, notably the Project Manager and the Object Repository.projects

¹² x-ref change to correct chapter # reference. Old reference was Chapt. 2.