CodeWarrior® Scripting Reference

Because of last-minute changes to CodeWarrior, some of the information in this manual may be inaccurate. Please read the Release Notes on the CodeWarrior CD for the latest up-to-date information.

Revised: 990518-?

How to Contact Metrowerks:

U.S.A. and international	Metrowerks Corporation 9801 Metric, Suite #100 Austin, TX 78758 U.S.A.
Canada	Metrowerks Inc. 1500 du College, Suite 300 Ville St-Laurent, QC Canada H4L 5G6
Ordering	Voice: (800) 377–5416 Fax: (512) 873–4901
World Wide Web	`http://www.metrowerks.com`
Registration information	`register@metrowerks.com`
Desktop/Games Technical support	`cw_support@metrowerks.com`
Embedded Technical support	`cw_emb_support@metrowerks.com`
Sales, marketing, & licensing	`sales@metrowerks.com`
International Ordering	Voice: +1 512 873 4724 Fax: +1 512 873 4901
Intl. Sales, marketing, & licensing	`intlsls@metrowerks.com`

Table of Contents

1

Scripting CodeWarrior

This manual details how to use the facilities available on your operating system to automate (script) the CodeWarrior IDE. Using a scripting language available to you, it is possible to make the IDE compile files, create files, build projects, etc all from a scripting language.

NOTE: This is a new manual. If you have inputs on how to make it better, send email to `wordwarrior@metrowerks.com` with your suggestions.

Overview of CodeWarrior IDE Scripting

The CodeWarrior IDE allows you to use a scripting language to control script operations. Depending on whether you are using the Mac OS or Microsoft Windows with CodeWarrior, you will want to refer to different parts of this manual.

Scripting CodeWarrior Using Applescript

To learn how to use AppleScript to control the CodeWarrior IDE, refer to the chapter in this manual entitled "Mac OS CodeWarrior Scripting" on page 7.

Scripting CodeWarrior on Microsoft Windows

To learn how to use script commands to control the CodeWarrior IDE, refer to the chapter in this manual entitled "CodeWarrior Scripting on Microsoft Windows" on page 85.

2

Mac OS CodeWarrior Scripting

This chapter introduces and discusses the Apple Event and AppleScript support provided by the CodeWarrior IDE.

CodeWarrior Apple Events Overview

This chapter discusses the AppleScript and Apple Event commands and classes supported in CodeWarrior. You should read this chapter if you would like to enhance and extend the capabilities of the CodeWarrior IDE.

The CodeWarrior IDE supports Apple Events. By scripting these Apple Events using AppleScript or another scripting editor, such as Frontier, it is possible to execute many CodeWarrior IDE commands without using the IDE directly. Scripting the CodeWarrior IDE is a way to automate repetitive tasks that do not need user interaction.There are many exciting things that you can do with AppleScript to harness the power of the IDE, such as automate builds, generate files automatically, and configure settings.

If you are primarily interested in writing scripts that manipulate and automate the IDE, then you are probably most interested in using AppleScript to put together an ensemble of Apple Events. If you would like to write program code to drive the CodeWarrior IDE from within your own computer program or tools, then you are probably most interested in the lower-levels of Apple Events, and not in AppleScript. This chapter is oriented toward working with AppleScript, but there is a discussion of low-level Apple Event coding in “Coding with CodeWarrior IDE and Apple Events” on page 83.

TIP: Look at the AppleScripts in the `(Scripts)` folder of the `Metrowerks CodeWarrior` folder for lots of cool AppleScripts. Reviewing these scripts will save you time when learning to write your own.

This chapter is not a tutorial. If you want to learn how to edit, save, and run AppleScripts, you will not find the information here. Instead, refer to other tools and sources of information listed in “AppleScript Tools and Reference Material” on page 8 for more information.

The topics in this chapter are:

TIP: You can run AppleScripts from within the CodeWarrior IDE. To learn about how to do this, read the section of the IDE User Guide manual that discusses this topic.

AppleScript Tools and Reference Material

You can find the tools provided by Apple Computer for editing and running AppleScripts on the CodeWarrior Reference CD in the `MacOS System Extensions` folder. Use the installer provided there to install the tools on your system.

Other editing and debugging tools are available from third-party vendors. These products are worth evaluating if you are going to do much AppleScripting. The following is neither an exhaustive list nor an endorsement of these products.

- Script Debugger (Late Night Software)
- Scripter (Main Event Software)

If you are a subscriber to the Apple Developer CD program or Apple Developer Mailing, you will find good information on Apple Events and AppleScripting on the CDs.

For more information on using and writing AppleScripts, you may want to consult other publications, such as:

- *AppleScript Language Guide: English Dialect* (Addison-Wesley)
- *Danny Goodman's AppleScript Handbook* (Random House)
- *The Tao of AppleScript* (Hayden Books)
- *Applied Mac Scripting* (M & T Books)

For information on more advanced topics such as writing your own Scripting Additions, or how to use the standard Scripting Additions, refer to *AppleScript Scripting Additions Guide* (Apple Computer).

On the internet, Apple Computer maintains a web site for AppleScript issues, as well as email lists of AppleScript topics. Point your web browser at `http://AppleScript.apple.com` to learn more.

Finally, for documentation on using low-level AppleEvents in program code, refer to *Inside Macintosh: Interapplication Communication* (Addison-Wesley).

Writing Your First CodeWarrior IDE AppleScript

To get started with AppleScript and the CodeWarrior IDE, let's take a look at a simple script that opens the IDE, brings it to the foreground on the Mac, opens a project, removes the binaries, and starts a build of the project. This script, shown in Listing A.1, is something that could be double-clicked to automatically do all these operations unattended.

Listing 0.1 My First CodeWarrior AppleScript

```
tell application "CodeWarrior IDE 4.0" (* go! *)
  activate (* bring CW to the front *)
  open file "SD:MyProj:MyProject.68K.mcp"
  Remove Binaries
  Make Project
end tell
```

You can imagine how convenient it will be to automate many tasks with AppleScript from this short example. Try entering this script in your Editor, such as Apple's Script Editor that comes with the AppleScript 1.1 software on the CodeWarrior Reference CD, and get it to run.

After getting this short example to run, you will probably be motivated to try some more extensive examples.

CodeWarrior IDE AppleScript Events

In general, Apple Events are grouped in categories or "suites" of events that provide some common theme for the events. There is a "Required" suite of events that includes open, print, quit and run. All scriptable applications should support the required suite. There are other suites of events defined in the *Apple Event Registry* document. In addition, there are other suites of events that are application-specific.

For many of the things that you probably want to do with the CodeWarrior IDE, it really isn't a concern which suite an event is from most of the time. However, you can view the "dictionary" of Apple Events that an application supports using the Open Dictionary command of your Script Editor. See the documentation that came with your editor for information about viewing the dictionary.

In this section, we discuss how to handle errors in AppleScript, and several categories of events that you can use to control the CodeWarrior IDE.

- Processing Errors

- Required Events
- File Handling Events
- Building Events
- Status/Query Events
- Navigation Events

Parameters

Some Apple Events listed in this section require a parameter called *filename-list*. The *filename-list* represents a single filename or a list of filenames and/or aliases. A single filename is a quoted character string. A list of filenames is enclosed in braces, {}, with the filenames separated by commas.

Listing 0.2 Example values for *filename*

```
"myprogram.c"
{"startup.p", "printout.p", "drawbox.p"}
{"codechecker.cpp"}
{"HD:CodeWarrior ƒ:My Projects:hello.c"}
file "myprogram.c"
alias "myprogram.c"
```

Processing Errors

When an AppleEvent is sent to CodeWarrior, errors may be returned to the script. Errors are not always evil occurrences, as sometimes you will want to trap errors to make your script do other things in response to the current conditions. For example, you can trap a file-not-found error and direct the CodeWarrior IDE to perform an alternate action as a result. Errors can be generated from the operating system, or from the application you're trying to script. Errors for the operating system are documented in Appendix C of the *AppleScript Language Guide*. Errors generated by the CodeWarrior IDE are documented here.

Errors are usually returned through the normal Error-return channel. However, for events that process a list of files, the errors are returned in `the result` (a built-in AppleScript variable). The list, with each member corresponding to an input file, is returned as the event's result with each list member.

In addition to operating system errors, such as out of memory errors, the error codes listed in Table A.1 may also be returned.

Table 0.1 CodeWarrior keyAEResult result codes (typeShortInteger)

Name	Value
`noErr`	`0`
`errShell_ActionFailed`	`1`
`errShell_FileNotFound`	`2`
`errShell_DuplicateFile`	`3`
`errShell_CompileError`	`4`
`errShell_MakeFailed` (compile or link error)	`5`
`errShell_NoOpenProject`	`6`
`errShell_WindowNotOpen`	`7`
`errShell_SegmentNotFound`	`8`

The result parameter, `keyAEResult`, is not set if there is an error while interpreting the AppleEvent (running out of memory, supplying a bad parameter type, and so on). In such cases, an error code is returned in the standard `keyErrorNumber` parameter.

Listing A.3 gives an example of an AppleScript that handles an error.

Listing 0.3 Error handling in AppleScript

```
try
  tell application "CodeWarrior IDE 4.0"
    set doclist to (Get Open Documents)
  end tell
```

```
on error number errnum
  display dialog "Bummer!" & errnum
end try
```

To see more examples of error handling in scripts, review some of the scripts in the `(Scripts)` folder in the same folder as your CodeWarrior IDE application.

Required Events

There are four events that are required for every application that claims to be AppleScriptable. This section discusses these four events and their syntax.

The events covered in this section are:

- Open
- Print
- Quit
- Run

Open

Purpose This event tells the CodeWarrior IDE to open the specified files.

```
Open filename-list [ converting ] expression
```

If `converting` is specified, any project files that were created with previous versions of the CodeWarrior IDE will be updated.

Listing 0.4 Example for Open

```
Open "HD:MyProject.mcp" converting yes
Open "HD:MyProject.mcp"
```

Print

Purpose This event tells the CodeWarrior IDE to print the specified files.

`Print` *filename-list*

Quit

Purpose This event tells the CodeWarrior IDE to quit.

Run

Purpose This event is sent to an application when it is double-clicked. Upon receiving the event, the application should launch itself.

File Handling Events

You will want to use the File Handling Apple Events of the CodeWarrior IDE to do things like add and remove files in a project, close a window, create and close a project, and save copies of files.

Here are the events covered in this section:

- Add Files
- Close Project
- Close Window
- Create Project
- Remove Files
- Save Error Window As
- Select
- Set Modification Date

Add Files

Purpose Adds the specified files to the current project.

`Add Files` *filename-list* `[ to segment` *number* `]`

Description This event is equivalent to the **Add Files** command in the **Project Menu**. The *filename-list* parameter describes a single filename or list of filenames to add the current project. The optional `to segment` parameter specifies the segment in the project in which to add the files. Replace *number* with the segment number to place the files in. The default is to create a new segment.

Returns A list of errors. The result code for each file added to the project can either return the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_DuplicateFile`
- `errShell_NoOpenProject`

Listing 0.5 Examples for Add Files

```
Add Files "MyFile.c"
Add Files "MyFile.c" to segment 2
Add Files {"MyFile.c", "MyFile2.c"}
Add Files {"MyFile.c", "MyFile2.c"} to segment 3
```

Close Project

Purpose Closes the current project.

```
Close Project
```

Returns None.

Close Window

Purpose Closes editor windows.

`Close Window` *filename* `[ saving` *status* `]`

Description This event is equivalent to the **Close** command in the **File Menu**. If *filename* is a string, `Close Window` first tries to find the first matching window name, searching front to back. If no window name matches *filename*, then `Close Window` causes a search for a matching filename.

To specify read/only windows, add " [r/o 1]" to the end of *filename*.

```
Close Window "hello.c [r/o 1]"
```

Table A.2 lists file saving options used with the close window AppleScript command.

Table 0.2 Saving options ('savo')

Name	Property Code
`yes` (Save changes)	`'yes '`
`no` (Do not save changes)	`'no  '`
`ask` (Ask the user whether to save)	`'ask '`

The optional `saving` parameter determines if the windows contents are saved before closing the window.

- If *status* is `yes`, save the window's contents.
- If *status* is `no`, discard changes made to the file.
- If *status* is `ask`, prompt the user whether or not to save the file.

Returns None.

Listing 0.6 Example for Close Window

```
Close Window "untitled"
    -- Closes first "untitled" window.
Close Window "hello.c" saving yes
Close Window "main.c [r/o 1]"
    -- Closes read/only window.
```

Create Project

Purpose Creates a new project file.

`Create Project` *filename* `[ from stationery ]`*alias*

Description Performs the **New Project** command in the **File Menu**. Replace *filename* with a single filename for the new project. If `from stationery` is specified, project stationery specified by *alias* is used to create the new project.

Returns The result code can have the following values:

- `noErr`
- `errShell_ActionFailed`

Listing 0.7 Examples for Create Project

```
Create Project "HardDisk:Projects:MyProject.µ"
Create Project "Foobe" from ¬
  stationery "HD:Dev:Metrowerks" & ¬
  "CodeWarrior:(Project Stationery):MacOS:C/C++:" & ¬
  "Basic Toolbox 68k:Basic Toolbox 68k.mcp"
Create Project "::sample project:sample.mcp"
```

Remove Files

Purpose Removes the specified file(s) from the current project.

`Remove Files` *filename*

Description Performs the equivalent of the **Remove Selected Items** command in the **Project Menu**. Replace *filename* with a single file or a list of files to remove from the current project.

Returns A list of errors. The result code can have one of the following values:

- noErr
- errShell_FileNotFound

- errShell_NoOpenProject

Listing 0.8 Examples for Remove Files

```
Remove Files "MyFile.c"
Remove Files { "MyFile.c", "YourFile.c" }
```

Save Error Window As

Purpose Saves the contents of the message window as a text file.

```
Save Error Window as filename
```

Description Performs the equivalent of the **Save A Copy As** command in the **File Menu** when the Message window is active. The contents of the Message window are saved with the name *filename*.

Returns None.

Listing 0.9 Example for Save Error Window As

```
Precompile "main.pch"
Save Error Window As "main.pch results"
```

Select

Purpose Selects an object from an open document in the CodeWarrior Editor.

```
Select reference
```

Description The parameter *reference* is the object to select.

Listing 0.10 Example for Select

```
Select text from character 5 to character 10 ¬
of line 8 of document 1
```

Set Modification Date

Purpose Sets the modification date of the specified file(s).

```
Set Modification Date filename-list to date
```

Returns A list of results of type short integer, or, if the `ExternalEditor` option is specified, a list of records of type `'ErrM'`.

Building Events

Here are the events discussed in this section:

- Build
- Check
- Check Syntax
- Compile
- Compile File
- Disassemble File
- Make Project
- Precompile
- Preprocess
- Remove Binaries
- Remove Object Code
- Run
- Run Project
- Touch
- Update Project

Build

Purpose Builds the current target or project.

```
Build
```

Description Performs the **Make** command in the **Project Menu**.

Returns By default, `Build` returns nothing.

Check

Purpose Checks the syntax of the specified file(s).

`Check`*filename-list*

Description This event is equivalent to performing the **Check Syntax** command in the **Project Menu**. Replace *filename-list* with a single filename or a list of filenames in the project.

By default, `Check` returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

Returns A list of results of type short integer.

Listing 0.11 Examples for Check File Syntax

```
Check File Syntax "MyFile.c"
Check File Syntax {"MyFile.c", "YourFile.c"} with ExternalEditor
```

Check Syntax

Purpose Checks the syntax of the specified file(s).

`Check Syntax` *filename-list* `[with ExternalEditor]`

Description This event is equivalent to performing the **Check Syntax** command in the **Project Menu**. Replace *filename-list* with a single filename or a list of filenames in the project.

By default, `Check Syntax` returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

If the `ExternalEditor` option is used, the environment returns the Message window contents instead of the usual list of short integer results. The AppleEvent keyword for `ExternalEditor` is `'Errs'`. It takes a boolean parameter.

Returns A list of results of type short integer, or, if the `ExternalEditor` option is specified, a list of records of type `'ErrM'`.

Listing 0.12 Examples for Check Syntax

```
Check Syntax "MyFile.c"
Check Syntax {"MyFile.c", "YourFile.c"} with ExternalEditor
```

Compile

Purpose Compiles the specified file(s).

```
Compile filename-list [with ExternalEditor]
```

Description This event is equivalent to performing the **Compile** command on the **Project Menu**. Replace *filename-list* with a single filename or a list of filenames.

By default, `Compile` returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

If the `ExternalEditor` option is specified, the environment returns the Message window contents as a list of `'ErrM'` objects.

Returns A list of errors of type short integer or, if `ExternalEditor` is specified, of type `'ErrM'`.

Listing 0.13 Examples for Compile

```
Compile "MyFile.c"
Compile {"MyFile.c", "YourFile.c"}
```

Compile File

Purpose Compiles the specified file(s).

`Compile File` *filename-list*

Description This event is equivalent to performing the **Compile** command on the **Project Menu**. Replace *filename-list* with a single filename or a list of filenames.

By default, `Compile` returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

Returns A list of errors of type short integer.

Listing 0.14 Examples for Compile

```
Compile File "MyFile.c"
Compile File {"MyFile.c", "YourFile.c"}
```

Disassemble File

Purpose Disassembles the specified file(s).

`Disassemble File` *filename-list*

Description This event is equivalent to performing the **Disassemble** command on the **Project Menu**. Replace *filename-list* with a single filename or a list of filenames.

Returns A list of errors of type short integer.

Listing 0.15 Examples for Disassemble File

```
Disassemble File "MyFile.c"
Disassemble File {"MyFile.c", "YourFile.c"}
```

Make Project

Purpose Makes the current project.

```
Make Project [with ExternalEditor]
```

Description Performs the **Make** command in the **Project Menu**.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Make Project` returns nothing. If `ExternalEditor` is specified, `Make Project` returns a list of errors of type `'ErrM'`.

Precompile

Purpose Precompiles the specified file.

```
Precompile source saving as destination [ with ExternalEditor ]
```

Description This event is equivalent to the **Precompile** command in the **Project Menu**. Replace *source* with the name of a file to precompile. Replace *destination* with the filename of the precompiled header.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Precompile` returns nothing. If `ExternalEditor` is specified, `Precompile` returns a list of errors of type `'ErrM'`.

Listing 0.16 Example for Precompile

```
Precompile "MyHeaders.pch" saving as "MyHeaders.mch"
Precompile "tip.pch" saving as "tip.mch" with ExternalEditor
```

Preprocess

Purpose Preprocesses the specified file.

```
Preprocess source[ with ExternalEditor ]
```

Description This event is equivalent to the **Preprocess** command in the **Project Menu**. Replace *source* with the name of a file to preprocess.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Preprocess` returns nothing. If `ExternalEditor` is specified, `Preprocess` returns a list of errors of type `'ErrM'`.

Listing 0.17 Examples for Preprocess

```
Preprocess "MyHeaders.c"
Preprocess "tip.c" with ExternalEditor
```

Remove Binaries

Purpose Removes the binary object code from the current project.

```
Remove Binaries
```

Description Performs the equivalent of the **Remove Object Code** command in the **Project Menu**.

Returns None.

Remove Object Code

Purpose Removes the binary object code from the current target or project.

```
Remove Object Code
```

Description Performs the equivalent of the **Remove Object Code** command in the **Project Menu**.

Returns None.

Run

Purpose Runs the current project or target.

```
Run
```

Description Performs the equivalent of the **Run** command in the **Project Menu**. This event builds then executes the current target if there are no compile or link errors.

Returns By default, `Run Project` returns nothing.

Run Project

Purpose Runs the current project

```
Run Project [ with ExternalEditor ] [ with SourceDebugger ]
```

Description Performs the equivalent of the **Run** command in the **Project Menu**. This event builds then executes the current project if there are no compile or link errors.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

If the `SourceDebugger` option is used, the environment launches the successfully-built project into the source-level debugger.

Returns By default, `Run Project` returns nothing. If `ExternalEditor` is specified, `Run Project` returns a list of errors that occurred when running the project, of type `'ErrM'`.

Listing 0.18 Examples for Run Project

```
Run Project with SourceDebugger
Run Project with ExternalEditor
```

Touch

Purpose Touches the specified file(s).

`Touch` *filename*

Description Performs the equivalent of clicking the **Touch** column in a Project window. Touching a file forces it to be recompiled during a make operation. Replace *filename* with a single file or a list of files to touch.

For more on touching a file to be recompiled, consult the *IDE User Guide* for information on synchronizing files.

Returns

A list of errors. Each result code can have one of the following values:

- `noErr`
- `errShell_FileNotFound`
- `errShell_NoOpenProject`

Listing 0.19 Examples for Touch

```
Touch "MyFile.c"
Touch { "MyFile.c", "YourFile.c" }
```

Update Project

Purpose

Updates the current project.

```
Update Project [ with ExternalEditor ]
```

Description

This command is equivalent to the **Bring Up To Date** command in the **Project Menu**. If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns

By default, `Update Project` returns nothing. If `ExternalEditor` is specified, `Update Project` returns a list of errors of type `'ErrM'`.

Status/Query Events

Here are the events discussed in this section:

- Get
- Set
- Get Definition
- Get Member Function Names
- Get Nonsimple Classes
- Get Open Documents
- Get Preferences

Get

Purpose Gets the object referenced.

```
Get reference [ as list of typeclass ]
```

Description The parameter *reference* is the object whose data is to be returned. The parameter *typeclass* is the desired types for the data, in order of preference.

Set

Purpose Sets the object referenced.

```
Set reference to anything
```

Description The parameter *reference* is the object whose data is to be changed. The parameter *anything* is the new value for the object.

Listing 0.20 Example for Set

```
tell application "CodeWarrior IDE 4.0" to
set numClasses to the count of classes
```

Get Definition

Purpose Queries the location(s) of a globally-scoped function or data object for the current project.

```
Get Definition string
```

Description The *string* is the name of the symbol you are interested in.

Returns Record containing a list of the function information.

Listing 0.21 Example for Get Definition

```
Get Definition "main"
```

Get Member Function Names

Purpose Gets a list of all the member functions of a class object.

```
Get Member Function Names reference
```

Returns List containing the information.

Listing 0.22 Example for Get Member Function Names

```
Get Member Function Names class "CPowerTelnetApp"
```

Get Nonsimple Classes

Purpose Gets a list of all the member functions of a class object.

```
Get Nonsimple Classes
```

Returns List containing the information.

Listing 0.23 Example for Get Nonsimple Classes

```
Get Nonsimple Classes class "CPowerTelnetApp"
```

Get Open Documents

Purpose Gets the list of open documents.

```
Get Open Documents
```

Returns List of documents in records of type `'docu'`. See Table A.40 for more information

Get Preferences

Purpose Gets settings from a panel.

```
Get Preferences [ of pref-list ] from panel panel-name
```

Description The *panel-name* must be the name of the preference panel file and not the name that appears in the preferences window. For example, to set C/C++ Language options, use `"C/C++ Compiler"` as the *panel-name* and not `"C/C++ Language"`.

Returns Record containing a list of the requested preferences. If you do not include *pref-list*, it returns all the preferences for *panel-name*.

Listing 0.24 Examples for Get Preferences

```
Get Preferences from panel "C/C++ Compiler"
Get Preferences of {File Name, SIZE Flags} ¬
  from panel "PPC Project"
```

Set Preferences

Purpose Specifies the settings for a panel.

`Set Preferences of panel` *panel-name* `to` *record*

Description Performs the equivalent of setting options using either the Preferences or Settings windows. This event lets you set the properties of the current project. It is not necessary to specify every preference, those not mentioned in the record retain their settings. The properties for different panels are listed in various tables from Table A.3 to Table A.38.

The *panel-name* must be the name of the preference panel file and not the name that appears in the preferences window. For example, to set C/C++ Language options, set *panel-name* to `"C/C++ Compiler"` and not `"C/C++ Language"`.

Returns None.

Listing 0.25 Examples for Set Preferences

```
Set Preferences of panel "PPC Project" to { ¬
  File Name:"MyProgram", File Creator:"Mine", SIZE Flags:23008 ¬
}
Set Preferences of panel "C/C++ Compiler" to { ¬
  Prefix File: "MacHeaders", ¬
  Activate CPlusPlus: TRUE, ¬
  Require Function Prototypes: FALSE ¬
}
Set Preferences of panel "C/C++ Warnings" to { ¬
  Extended Error Checking: TRUE ¬
}
```

Get Project File

Purpose Gets information on a project entry.

`Get Project File` *file-number* `segment` *seg-number*

Returns The information of the specified entry in the current project as a record of type `'SrcF'`. The *file-number* parameter specifies a file within its segment or group. The *seg-number* parameter specifies a segment or group within the project. Numbering for both parameters are short integers beginning at 1.

Listing 0.26 Examples for Get Project File

```
get project file 1 segment 1
    -- First entry in project
get project file 1 segment 2
    -- First entry in 2nd segment
```

Set Project File

Purpose Sets information on a project entry.

```
Set Project File filename to record
```

Description Changes the settings for the specified entry in the open project. The *record* parameter is of type `'SrcF'`. Only the symbols and weak link fields are allowed. Listing A.28 shows how to set weak linking for a library `InterfaceLib` that is in a project.

Returns None.

Listing 0.27 Example for Set Project File

```
Set Project File "InterfaceLib" to {weak link: true}
```

Set Current Target

Purpose Sets the current target for a project.

```
Set Current Target name-of-target
```

Description This AppleEvent causes the build target to change. This event would be useful when changing the build target in a project, so that a new target can be built or otherwise operated on with other AppleEvents. To learn more information about setting the current build target, refer to the *IDE User Guide*.

Returns None.

Listing 0.28 Example for Set Current Target

```
Set Current Target "Muscle 68K"
```

Set Default Project

Purpose Sets the default project.

```
Set Default Project name-of-target
```

Description This AppleEvent causes the default project to change. To learn more information about setting a default project, refer to the *IDE User Guide*.

Returns None.

Listing 0.29 Example for Set Default Project

```
Set Current Project "Muscle.mcp"
```

Get Project Specifier

Purpose Gets the filename of the project.

```
Get Project Specifier
```

Returns The name of the current project.

Get Segments

Purpose Gets the descriptions of all segments/groups in the open project.

```
Get Segments
```

Returns List of documents in records of type `'Seg '`. Refer to “Segment” on page 47 for more information

Set Segment

Purpose Sets preferences for the current project.

```
Set Segment number to record
```

Description Sets information for a segment or group in the open project. Segment numbering starts at 1. The *record* parameter is an object of type `'Seg '`. Listing A.31 shows how to rename a segment/group in a project. Refer to “Segment” on page 47 for more information.

Returns None.

Listing 0.30 Example for Set Segment

```
Set Segment 1 to {name:"New Sources"}
```

Is In Project

Purpose Are the specified file(s) are in the project?

```
Is In Project filename
```

Description Replaces *filename* with a single filename or a list of filenames.

Returns A list of errors. The result code for each specified file can have the following values:

- `noErr` if the file is in the project
- `errShell_FileNotFound` if file is not in the project.

Listing 0.31 Examples for Is In Project

```
Is In Project "SillyBalls.c"
Is In Project { "SillyBalls.c", "Initialize.c" }
```

Reset File Paths

Purpose Resets access paths for all files belonging to the open project.

```
Reset File Paths
```

Returns None.

Close

Purpose Closes an object. The `saving` and `saving in` parameters are optional, and have the range of values listed here.

```
Close reference [ saving yes/no/ask ] [ saving in alias ]
```

Returns None.

Count

Purpose Counts the number of elements within an object.

```
Count reference each type class
```

Returns An integer indicating the number of elements.

Make

Purpose Makes a new element. The `as list of`, `as`, `with data`, and `with properties` parameters are optional, and have the range of values listed here.

```
Make new [ as list of type class ] [ at location reference ] [
with data anything ] [ with properties record ]
```

Returns A reference to the new object(s).

Navigation Events

Here are the events discussed in this section:

- Goto Function
- Goto Line
- Open Browser

Goto Function

Purpose Jumps to the specified function defined in the active editor window.

```
Goto Function name
```

Description This event is equivalent to selecting a routine name in the active editor window’s function pop-up menu. The insertion point does not move when a function is selected with this event.

Returns None.

Listing 0.32 Examples for Goto Function

```
Goto Function "main"
Goto Function "SkipBlanks"
```

Goto Line

Purpose Jumps to the specified line number in the active editor window.

```
Goto Line number
```

Description `Goto Line` moves the insertion point to the specified line number, *number*, in the active editor window. If the line number specified exceeds the last line number, the insertion point is placed at the last line.

Returns None.

Listing 0.33 Examples for Goto Line

```
Goto Line 1
Goto Line 493
```

Open Browser

Purpose Displays a class, member function, or data member object in a single class browser window. You cannot display a procedural function.

```
Open Browser reference
```

Returns None.

Listing 0.34 Example for Open Browser

```
Open Browser class "CPowerTelnetApp"
Open Browser member function 2 of class "CPowerTelnetApp"
```

CodeWarrior IDE AppleScript Classes

CodeWarrior events have several classes to let you control the CodeWarrior IDE's actions and option settings. These classes are based on the items available in the **Preferences** and **Target Settings** windows.

The available classes are determined by the compiler (C/C++, Java, or Pascal) and the processor for which it is generating object code (68k-based, Intel x86 or PowerPC-based Macintosh). In other words, the preference classes available for the CodeWarrior C/C++ compiler that generates object code for the PowerPC are different from those available for the CodeWarrior C compiler that generates 68K object code.

AppleScript classes for the CodeWarrior environment are, for the most part, separated by preference panels, project settings, and then by miscellaneous environment items.

- Project Classes—contains properties for each aspect of a project, from target parameters to final application settings.
- Compiler Classes—contains properties for each language and compiler, to configure compilation settings and warnings.
- CodeGen Classes—contains properties for each possible kind of code that can be generated by the CodeWarrior compilers.
- Disassembler Classes—contains properties for the disassemblers.
- Linker Classes—contains properties for the linker settings.
- Build Classes—contains properties for build environment settings and errors.
- Browser Classes—contains properties for the code browser.
- Editor Classes—contains properties for the CodeWarrior IDE text editor.
- Object Classes—contains properties that describe objects, including data members, classes and base classes, and member functions.
- Misc Classes—contains properties that describe miscellaneous aspects of the CodeWarrior IDE environment.

Many options that use a pop-up menu in the preference panel now use an enumerated type to specify their values, instead of an integer. For example, to set the Code Model, you should now use small, smart, or large, instead of 1, 2, or 3.

WARNING! Your script will not work if you use integer to set a property that expects a symbol.

Project Classes

- 68K Project
- PowerPC Project

- Java Project
- Win32/x86 Project
- Access Paths
- Path Information
- Target Settings
- File Mapping Information
- Segment
- Project File

68K Project

Table A.3 lists the 68K Project class properties.

Table 0.3 68K Project Class

Name	Property Type
`Project Type`	• `standard application` • `CFM68K application` • `code resource` • `library` • `shared library` • `MPW Too` • `Pilot Application` • `Pilot Code Resource`
`File Name`	`string`
`File Creator`	`string`
`File Type`	`string`
`Minimum Size`	`integer`
`Preferred Size`	`integer`

Name	Property Type
`SIZE Flags`	`small integer` (`SIZE` flag bits must be computed as an integer value)
`SYM File`	`string`
`Resource Name`	`string`
`Display Dialogs`	`boolean`
`Merge To File`	`boolean`
`Resource Flags`	`small integer`
`Resource Type`	`string`
`Resource ID`	`small integer`
`Multi Segment`	`boolean`
`Library Type`	• `A4 relative` • `A5 relative` • `CFM68K` • `Pilot Library`
`Seg Type`	`string`
`Stack Size`	`integer`
`Start-up Code`	• `standard` • `The Debugger Aware` • `custom`
`Header Type`	• `standard` • `device driver` • `desk accessory` • `custom`
`RSEG Application`	`boolean`

PowerPC Project

Table A.4 lists the PPC Project class properties.

Table 0.4 PPC Project Class

Name	Property Type
`Project Type`	• `standard application` • `code resource` • `library` • `shared library`
`File Name`	`string`
`File Creator`	`string`
`File Type`	`string`
`Minimum Size`	`integer`
`Preferred Size`	`integer`
`SIZE Flags`	`small integer` (`SIZE` flag bits must be computed as an integer value)
`SYM File`	`string`
`Resource Name`	`string`
`Display Dialogs`	`boolean`
`Merge To File`	`boolean`
`Resource Flags`	`small integer`
`Resource Type`	`string`
`Resource ID`	`small integer`
`Stack Size`	`integer`
`Header Type`	• `none` • `native`

Java Project

Table A.5 lists the Java Project class properties.

Table 0.5 Java Project Class

Name	Property Type
`Main Class`	`string`
`Java Project Type`	• `java applet` • `java application` • `java library`
`Arguments`	`string`
`Compress`	`boolean`
`HTML Helper App`	`string`

Win32/x86 Project

Table A.6 lists the x86 Project class properties.

Table 0.6 x86 Project Class

Name	Property Type
`Project Type`	• `standard application` • `shared library` • `library`
`File Name`	`string`
`Min Heap Size`	`integer`
`Preferred Heap Size`	`integer`
`Base Address`	`integer`
`Max Stack Size`	`integer`
`Min Stack Size`	`integer`

Access Paths

Table A.7 lists the properties for the Access Paths Class.

Table 0.7 Access Paths Class

Name	Property Type
`User Paths`	`list` (of path information records)
`Always Full Search`	`boolean`
`Convert Paths`	`boolean`
`System Paths`	`list` (of path information records)

Path Information

A path information record may contain the properties shown in Table A.8. It must contain at least the name field.

Table 0.8 Path Information Class

Name	Property Type
`name`	`string`
`recursive`	`boolean`
`origin`	• `absolute` • `project relative` • `shell relative` • `system relative`
`host flags`	• `1` (for the Mac OS) • `2` (for Windows)

If you use a string instead of a path information record, CodeWarrior sets recursive to true and origin to project relative.

To clear all the access path entries listed in the Access Paths preference panel, set the User Paths or System Paths property to an empty list. For example, in AppleScript, this statement removes all entries, including the default entries, `{Project ƒ}` and `{Compiler ƒ}`:

```
Set Preferences of panel "Access Paths" to ¬
  {User Paths: {}, System Paths: {} }
```

To add a default entry back, add an access path record with the name set to ":" and with the origin set to project relative (for `{Project ƒ}`) or shell relative (for `{Compiler ƒ}`). For example, this statement sets User Paths to `{Project ƒ}` and System Paths to `{Compiler ƒ}`:

```
Set Preferences of panel "Access Paths" to ¬
  {User Paths:   {{name: ":", ¬
                   origin: project relative}},¬
   System Paths: {{name: ":", ¬
                   origin: shell relative}}}
```

For more information on the meaning of the properties listed in Table A.7 and Table A.8, see “Access Paths” on page 43.

Target Settings

You set the current target parameters in the **Target Settings** options panel with the properties shown in Table A.9.

Table 0.9 Target Settings Class

Name	Property Type
`Target Name`	`string`
`Linker`	`string`
`Post Linker`	`string`
`Output Directory Path`	`string`

Name	Property Type
`Output Directory Origin`	• `absolute` • `project relative` • `shell relative` • `system relative`
`Pre Linker`	`string`

The `Target Name` string is the name of the target (you choose this name). The `Linker` string must be the name of one of the files in the `Linkers` folder of the `CodeWarrior Plugins` folder. The `Post Linker` string must be the name of one of the files in the `Post Linkers` folder of the `CodeWarrior Plugins` folder. The `Output Directory Path` string is a string that points to a location on your hard disk where the output files should be placed after linking. You can make this path absolute, or you can make it relative to the location of the project (project relative), compiler (compiler relative), or system (system relative).

The following is a list of the names of the linkers included with CodeWarrior:

- `MacOS 68K Linker`
- `MacOS PPC Linker`
- `MacOS Merge`
- `Java Linker`
- `MW JavaDoc Linker`
- `Win32 x86 Linker`

For example, the following statement changes the current target to Macintosh 68K:

```
Set Preferences of panel "Target Settings" to ¬
  {Linker: "MacOS 68K Linker"}
```

File Mapping Information

The File Mapping Information is a list of all the types of files you can include in the current project. It contains records described in Table A.10.

Table 0.10 File Mapping Information Class

Name	Property Type
`File Type`	`string`
`Extension`	`string`
`Precompiled`	`boolean`
`Resource File`	`boolean`
`Launchable`	`boolean`
`Ignored by Make`	`boolean`
`Compiler`	`string`

The `Compiler` string must be the name of one of the files in the `Compilers` folder of the `CodeWarrior Plugins` folder. These names are different from the names that appear in the Compiler pop-up menu of the Target preference panel. Table A.11 shows you which name to use for the compilers included with CodeWarrior.

Table 0.11 Choosing a compiler

To target...	with...	specify this string.
`68K Macintosh`	`Metrowerks C/C++`	`"MW C/C++ 68K"`
	`Metrowerks Pascal`	`"MW Pascal 68K"`
	`Rez`	`"Rez"`
	`Library Importer`	`"Lib Import 68K"`
	`MPW .o Importer`	`"MPW Import 68K"`
	`PEF Importer`	`"PEF Import 68K"`

To target...	with...	specify this string.
`Power Macintosh`	`Metrowerks C/C++`	`"MW C/C++ PPC"`
	`Metrowerks Pascal`	`"MW Pascal PPC"`
	`Rez`	`"Rez"`
	`Library Importer`	`"Lib Import PPC"`
	`PEF Importer`	`"PEF Import PPC"`
	`XCOFF Importer`	`"XCOFF Import PPC"`
`Win32/x86`	`Metrowerks C/C++`	`"MW C/C++ x86"`
	`Resource Compiler`	`"MW WinRC"`
	`Resource Importer`	`"WinRes Import"`
	`x86 Lib Importer`	`"Lib Import x86"`
	`x86 Obj Import`	`"Obj Import x86"`
`Java`	`Java`	`"MW Java"`

To specify that a file isn't compiled, use the empty string `""` for the compiler. For example, these statements show how to add an entry for text files that end in `.txt` and are not compiled.

```
set currPrefs to Get Preferences from panel "Target"
set Mappings of currPrefs to ¬
  Mappings of currPrefs & ¬
  {{File Type:"TEXT", Extension:".txt", ¬
    Compiler:"", Precompiled:false, ¬
    Resource File:false, Launchable:true, ¬
    Ignored by Make:true}}
Set Preferences of panel "Target" to currPrefs
```

Segment

The Segment Class properties, as shown in Table A.12, contain information about a segment or group in the open project,

Table 0.12 Segment Class

Name	Property Type
`name`	`string`
`filecount` (read only)	`small integer`
`seg-preloaded` (68K only)	`boolean`
`seg-protected` (68K only)	`boolean`
`seg-locked` (68K only)	`boolean`
`seg-purgeable` (68K only)	`boolean`
`seg-system heap` (68K only)	`boolean`

Project File

The Project File Class contains information about an entry in a project file. Table A.13 illustrates the available properties.

Table 0.13 Project File Class

Name	Property Type
`filetype` (read only)	• `source` • `unknown`
`name` (read only)	`string`
`disk file` (read only)	`file specification`
`codesize` (read only)	`integer`
`datasize` (read only)	`integer`
`up to date` (read only)	`boolean`

Name	Property Type
`symbols`	`boolean`
`initialize before`	`boolean`
`includes` (read only)	`file specification`
`weak link` (PPC only)	`boolean`

Compiler Classes

- C/C++ Compiler
- Java Compiler
- Pascal Compiler
- Rez Resource Compiler
- Windows Resource Compiler

C/C++ Compiler

Table A.14 lists the CodeWarrior C/C++ Compiler class properties.

NOTE: In AppleScript, you must refer to the C/C++ Language preference panel as: `panel "C/C++ Compiler"`.

Table 0.14 C/C++ Compiler Class

Name	Property Type
`Prefix File`	`string`
`Activate CPlusPlus`	`boolean`
`ARM Conformance`	`boolean`
`ANSI Keywords Only`	`boolean`
`Require Function Prototypes`	`boolean`
`Expand Trigraph Sequences`	`boolean`

Name	Property Type
`Enums Always Ints`	`boolean`
`MPW Pointer Type Rules`	`boolean`
`Exception Handling`	`boolean`
`AutoInlining`	`boolean`
`Pool Strings`	`boolean`
`Dont Reuse Strings`	`boolean`
`ANSI Strict`	`boolean`
`MPW Newlines`	`boolean`
`RTTI`	`boolean`
`Multibyte Aware`	`boolean`
`Enable wchar_t`	`boolean`
`Use Unsigned Chars`	`boolean`
`ECPlusPlus Compatibility`	`boolean`
`Objective C`	`boolean`
`Inlining`	• `inline_none` • `inline_smart` • `inlinedepth_1` • `inlinedepth_2` • `inlinedepth_3` • `inlinedepth_4` • `inlinedepth_5` • `inlinedepth_6` • `inlinedepth_7` • `inlinedepth_8` • `inline_always`
`Enable bool Support`	`boolean`

Name	Property Type
`Direct To SOM`	• `SOMoff` • `SOMon` • `SOMonWithEnv`
`Deferred Inlining`	• `boolean`

Table A.15 lists the CodeWarrior C/C++ Warnings class properties.

Table 0.15 C/C++ Warnings Class

Name	Property Type
`Unused Variables`	`boolean`
`Inconsistent Class Struct`	`boolean`
`Unused Arguments`	`boolean`
`Illegal Pragmas`	`boolean`
`Empty Declarations`	`boolean`
`Possible Errors`	`boolean`
`Extra Commas`	`boolean`
`Extended Error Checking`	`boolean`
`Treat Warnings As Errors`	`boolean`
`Hidden Virtual Functions`	`boolean`
`Implicit Arithmetic` Conversions	`boolean`
`NonInlined Functions`	`boolean`

Java Compiler

Table A.16 lists the CodeWarrior Java Compiler class properties.

Table 0.16 Java Compiler Class

Name	Property Type
`Method Inlining`	`boolean`

Pascal Compiler

Table A.17 lists the Pascal Language Class options. (In AppleScript, you must refer to the Pascal Language preference panel as: `panel "Pascal Compiler"`)

Table 0.17 Pascal Compiler Class

Name	Property Type
`Activate Range Checking`	`boolean`
`Use Propagation`	`boolean`
`Activate Overflow Checking`	`boolean`
`Case Sensitive`	`boolean`
`ANS Conformance`	`boolean`
`Activate ObjectPascal`	`boolean`
`Strings copy using length byte`	`boolean`
`Pool Strings`	`boolean`
`Dont Reuse Strings`	`boolean`
`Pool Sets`	`boolean`
`Dont Reuse Sets`	`boolean`
`Prefix File`	`string`
`Relax Pointer Compatibility`	`boolean`
`Optimize class hierarchy`	`boolean`
`Pointer based objects`	`boolean`

Name	**Property Type**
`Expand method tables`	`boolean`
`Inline method dispatching`	`boolean`
`Activate NilChecking`	`boolean`
`Trap Unmatched Cases`	`boolean`
`Copy Value Parameter`	`boolean`
`Turbo Pascal IO`	`small integer`

Table A.18 lists the Pascal Warnings class properties.

Table 0.18 Pascal Warnings Class

Name	**Property Type**
`Modified ForLoop Indexes`	`boolean`
`Function Returns`	`boolean`
`Undefined Routines`	`boolean`
`GotoAndLabels`	`boolean`
`BranchingIntoWith`	`boolean`
`BranchingIntoFor`	`boolean`
`BranchingBetweenCase`	`boolean`
`BranchingBetweenIfAndElse`	`boolean`
`Unused Variables`	`boolean`
`Unused Arguments`	`boolean`
`Check string param sizes`	`boolean`

Rez Resource Compiler

Table A.19 lists the properties for the CodeWarrior Rez Compiler Class.

Table 0.19 Rez Compiler Class

Name	Property Type
`Redeclared Types`	`boolean`
`RezPrefix File`	`string`
`Escape Control Chars`	`boolean`
`Max width`	`small integer`
`Filter Mode`	`Skip`, or `Only`
`Filtered Types`	`string`
`Alignment`	`small integer`
`Script Mode`	`Roman`, `Japanese`, `Korean`, `SimpChinese`, or `TradChinese`

Windows Resource Compiler

Table A.20 lists the properties for the Windows Resource Compiler Class.

Table 0.20 Windows Resource Compiler Class

Name	Property Type
`Prefix File`	`string`

CodeGen Classes

- 68K CodeGen
- PPC CodeGen
- IR Optimizer
- Win32/x86 CodeGen

68K CodeGen

Table A.21 lists the 68K Processor class properties. In AppleScript, you must refer to the 68K Processor preference panel as: `panel "68K CodeGen"`

Table 0.21 68K CodeGen Class

Name	Property Type
`Struct Alignment`	• `Align_68k` • `Align_68k_4byte` • `Align_PPC`
`Peephole Optimizer`	`boolean`
`CSE Optimizer`	`boolean`
`Optimize For Size`	`boolean`
`Use Profiler`	`boolean`
`Code Model`	• `small` • `smart` • `large`
`MC68020 CodeGen`	`boolean`
`Floating Point CodeGen`	• `SANE` • `MC68881` • `Library` • `PalmOS`
`Far Method Tables`	`boolean`
`Far String Constants`	`boolean`
`Four Bytes Ints`	`boolean`
`Eight Byte Double`	`boolean`
`Far Data`	`boolean`

Name	Property Type
`PC Relative Strings`	`boolean`
`MPW Calling Conventions`	`boolean`

PPC CodeGen

Table A.22 lists the PPC Processor preference panel class properties. In AppleScript, you must refer to the PPC Processor preference panel as: `panel "PPC CodeGen"`

Table 0.22 PPC CodeGen Class

Name	Property Type
`Struct Alignment`	• `Align_68k` • `Align_68k_4byte` • `Align_PPC`
`Processor`	• `PPC_Generic` • `PPC_601` • `PPC_603` • `PPC_603e` • `PPC_604` • `PPC_604e` • PPC_750
`Peephole Optimizer`	`boolean`
`Use Profiler`	`boolean`
`Make String ReadOnly`	`boolean`
`Schedule`	`boolean`
`Store Data in TOC`	`boolean`
`Use FMADD Instructions`	`boolean`
`Processor Specific`	`boolean`

Name	Property Type
`Traceback Tables`	• `TB_None` • `TB_Inline` • `TB_OutOfLine`
`Altivec`	• `boolean`

IR Optimizer

Table A.23 lists the IR Optimizer class properties.

Table 0.23 IR Optimizer Class

Name	Property Type
`Optimize Space`	`boolean`
`Optimize Speed`	`boolean`
`Common Subexpressions`	`boolean`
`Loop Invariants`	`boolean`
`Propagation`	`boolean`
`Dead Store Elimination`	`boolean`
`Strength Reduction`	`boolean`
`Dead Code Elimination`	`boolean`
`Lifetime Analysis`	`boolean`
`Optimizations Log`	`boolean`

Win32/x86 CodeGen

Table A.24 lists the x86 CodeGen class properties.

Table 0.24 Win32/x86 CodeGen Class

Name	Property Type
`Peephole Optimizer`	`boolean`
`Machine Code Listing`	`boolean`
`Byte Alignment`	`small integer`
`Sym Debug Information`	`boolean`
`CodeView Debug Info`	`boolean`
`Register Coloring`	`boolean`
`Expand Intrinsics`	`boolean`
`Disable Optimizations`	`boolean`
`Instruction Scheduling`	`boolean`
`Target Processor`	• `Generic X86` • `Pentium` • `Pentium Pro` • `Pentium II` • `AMD K6`
`Instruction Set`	• `None` • `MMX` • `MMX_K63D` • `K6 3D`

Disassembler Classes

- 68K Disassembler
- PowerPC Disassembler

68K Disassembler

Table A.25 lists the properties for the 68K Disassembly Class.

Table 0.25 68K Disassembler Class

Name	Property Type
`Show Code`	`boolean`
`Show Source`	`boolean`
`Dont show hex`	`boolean`
`Show Data`	`boolean`
`Show Exceptions`	`boolean`
`Show SYM`	`boolean`
`Show Names`	`boolean`

PowerPC Disassembler

Table A.26 lists the properties for the PowerPC Disassembly Class.

Table 0.26 PowerPC Disassembly Class

Name	Property Type
`Show Code`	`boolean`
`Show Source`	`boolean`
`Dont show hex`	`boolean`
`Show Data`	`boolean`
`Show Exceptions`	`boolean`
`Show SYM`	`boolean`
`Show Names`	`boolean`
`Use Extended Mnemonics`	`boolean`

Linker Classes

- 68K Linker
- CFM68K Linker

- Java Linker
- Mac OS Merge Linker
- PowerPC Linker
- PowerPC PEF Linker
- Win32/x86 Linker

68K Linker

Table A.27 lists the 68K Linker class properties.

Table 0.27 68K Linker Class

Name	Property Type
`Generate SYM File`	`boolean`
`Full Path In Sym Files`	`boolean`
`Generate Link Map`	`boolean`
`Fast Link`	`boolean`
`Suppress Warnings`	`boolean`
`MacsBug Symbols`	• `none` • `oldsymbols` • `newsymbols`
`Generate A6 Stack Frames`	`boolean`
`Link Single Segment`	`boolean`
`Merge Compiler Glue`	`boolean`
`Strip Static Init Code`	`boolean`

CFM68K Linker

Table A.28 lists the CFM68K Linker class properties.

Table 0.28 CFM68K Linker Class

Name	Property Type
`Export Symbols`	• `none` • `expfile` • `all` • `pragma`
`Old Definition`	`integer`
`Old Implementation`	`integer`
`Current Version`	`integer`
`Share Data Section`	`boolean`
`Expand Uninitialized Data`	`boolean`
`Fragment Name`	`string`
`Initialization Name`	`string`
`Main Name`	`string`
`Termination Name`	`string`
`Force Indirect Access`	`boolean`
`Far Data Threshold`	`integer`
`Global Data Alignment`	• `align1byte` • `align2byte` • `align4byte` • `align8byte`
`Library Folder ID`	`small integer`

Java Linker

Table A.29 lists the Java Linker class properties.

Table 0.29 Java Linker Class

Name	Property Type
`File Name`	`string`
`File Creator`	`string` Possible values include '`JAVA`' or '`MWZP`' as well as the creator types for Metrowerks Java or ClassWrangler.
`File Type`	`string` (this is 4 characters: '`ZIP `')
`Output Type`	• `zip file` • `runable zip file` • `droplet` • `folder`
`Compress Zip`	`boolean`

Mac OS Merge Linker

Table A.30 lists the Mac OS Merge Linker class properties.

Table 0.30 Mac OS Merge Linker Class

Name	Property Type
`Project Type`	`constant` (The type of the project)
`File Name`	`boolean`
`File Creator`	`string` (The creator type of the finished binary)

Name	Property Type
`File Type`	`string` (The file type of the finished binary)
`Suppress Warnings`	`boolean`
`Copy Fragments`	`boolean`
`Copy Resources`	`boolean`
`Skip Resource Types`	`string`

PowerPC Linker

Table A.31 lists the PPC Linker class properties.

Table 0.31 PPC Linker Class

Name	Property Type
`Generate SYM File`	`boolean`
`Full Path In Sym Files`	`boolean`
`Generate Link Map`	`boolean`
`Link Mode`	• `fast` • `normal` • `slow`
`Suppress Warnings`	`boolean`
`Initialization Name`	`string`
`Main Name`	`string`
`Termination Name`	`string`
`Strip Static Init Code`	`boolean`
`Duplicate Item Warning`	`boolean`

PowerPC PEF Linker

Table A.32 lists the PPC PEF class properties.

Table 0.32 PPC PEF Class

Name	Property Type
`Export Symbols`	• `none` • `expfile` • `all` • `pragma`
`Old Definition`	`integer`
`Old Implementation`	`integer`
`Current Version`	`integer`
`Code Sorting`	• `nosort` • `pragmas` • `depth` • `breadth` • `sortfile`
`Share Data Section`	`boolean`
`Expand Uninitialized Data`	`boolean`
`Fragment Name`	`string`
`Library Folder ID`	`small integer`
`Collapse Reloads`	`boolean`

Win32/x86 Linker

Table A.33 lists the x86 Linker class properties.

Table 0.33 x86 Linker Class

Name	Property Type
`Generate SYM File`	`boolean`
`Entry Point Usage`	• `none` • `default` • `user specified`
`Entry Point`	`string`
`SubSystem`	• `unknown` • `native` • `Windows GUI` • `Windows CUI`
`SubSystem Major Id`	`small integer`
`SubSystem Minor Id`	`small integer`
`User Major Id`	`small integer`
`User Minor Id`	`small integer`
`Generate Link Map`	`boolean`
`Generate CV Info`	`boolean`
`Command Line File`	`string`

Build Classes

- Build Extras
- Error Information

Build Extras

Table A.34 describes the properties for the Build Extras Class.

Table 0.34 Build Extras Class

Name	Property Type
`Browser active`	`boolean`
`Modification date caching`	`boolean`
`Dump Browser Info` (read only)	`boolean`
`Cache Subproject Data` (read only)	`boolean`

Error Information

This class describes a single error or warning from the compiler or the linker. This class is used by all compilers for all processors. The properties for this class are listed in Table A.35.

Table 0.35 Error Information Class

Name	Property Type
`messageKind` (read only)	• `information` • `compiler error` • `compiler warning` • `definition` • `linker error` • `linker warning` • `find result` • `generic error`
`message` (read only)	`string`
`disk file` (read only)	`file specification`
`line Number` (read only)	`integer`

Browser Classes

- Browser Coloring
- Browser Catalog
- Function Information

Browser Coloring

Table A.36 lists the **Browser Coloring** preference panel properties.

Table 0.36 Browser Coloring Class

Name	Property Type
`Browser Keywords`	`boolean`
`Classes Color`	`RGB values list`
`Constants Color`	`RGB values list`
`Enums Color`	`RGB values list`
`Functions Color`	`RGB values list`
`Globals Color`	`RGB values list`
`Macros Color`	`RGB values list`
`Templates Color`	`RGB values list`
`Typedefs Color`	`RGB values list`

Browser Catalog

The Browser Catalog Class elements may be referred to by numeric index, and by name.

Function Information

The Function Information class properties are described in Table A.37.

Table 0.37 Function Information Class Properties

Name	Property Type
`disk file` (read only)	`file specification`
`lineNumber` (read only)	`integer`

Editor Classes

- Editor
- Font
- Document
- Character
- Insertion Point
- Custom Keywords
- Line
- Text
- Selection-Object
- Syntax Coloring
- Window

Editor

Table A.38 lists the Editor class properties.

Table 0.38 Editor Class

Name	Property Type
`Remember window`	`boolean`
`Main Text Color`	`RGB values list`
`Background Color`	`RGB values list`
`Context Popup Delay`	`boolean`

Name	Property Type
`Remember selection`	`boolean`
`Use Drag & Drop Editing`	`boolean`
`Flash delay`	`integer`
`Dynamic scroll`	`boolean`
`Balance`	`boolean`
`Remember font`	`boolean`
`Sort Function Popup`	`boolean`
`Use Multiple Undo`	`boolean`
`Save on update`	`boolean`

An RGB values list is a list of three numbers from 0 to 65,535 that specifies how much red, green, and blue a color contains. For example, this example code sets the main text color to red.

```
set Prefs to Get Preferences from panel "Editor"
set Main Text Color of Prefs to {65535,0,0}
Set Preferences of panel "Editor" to Prefs
```

Font

Table A.39 lists the Font class properties.

Table 0.39 Font Class

Name	Property Type
`Auto Indent`	`boolean`
`Tab size`	`small integer`
`Text font`	`string`
`Text size`	`small integer`

Document

This class, shown in Table A.40, contains class properties for a text file opened with the CodeWarrior Editor. The plural form for this class should be referred to as Documents.

The elements for a document are:

- `character` by numeric index, before/after another element, as a range of elements, or satisfying a test
- `insertion point` before/after another element
- `line` by numeric index, as a range of elements, before/after another element
- `text` as a range of elements

Table 0.40 Document Class

Name	Property Type
`name` (read only)	`string`
`kind`	• `project` • `editor document` • `message` • `file compare` • `catalog document` • `class browser` • `single class browser` • `symbol browser` • `class hierarchy` • `single class hierarchy` • `project inspector` • `ToolServer worksheet` • `build progress document`

Name	Property Type
`file permissions` (read only)	• `read write` • `read only` • `checked out read write` • `checked out read only` • `checked out read modify` • `locked` • `none`
`location` (read only)	`file specification`
`index` (read only)	`integer`
`window` (read only)	`window`

Character

Table A.41 describes the Character class properties.

Table 0.41 Character Class Properties

Name	Property Type
`offset` (read only)	`integer`
`length` (read only)	`integer`

Insertion Point

Table A.42 describes the Insertion Point Class properties.

Table 0.42 Insertion Point Class Properties

Name	Property Type
`length` (read only)	`integer`
`offset` (read only)	`integer`

Custom Keywords

Table A.43 describes the Custom Keywords class properties.

Table 0.43 Custom Keywords Class Properties

Name	Property Type
`Custom color 1`	`RGB color values list`
`Custom color 2`	`RGB color values list`
`Custom color 3`	`RGB color values list`
`Custom color 4`	`RGB color values list`

Line

Table A.44 describes the properties for the Line class. The plural form of the class should be referred to as Lines. This class has elements that may be described as follows:

- `character` by numeric index, as a range of elements, and before/after another element

Table 0.44 Line Class Properties

Name	Property Type
`index` (read only)	`integer`
`offset` (read only)	`integer`
`length` (read only)	`integer`

Text

Table A.45 describes the Text class properties. The Text Class has the following elements:

- `character` by numeric index, before/after another element, as a range of elements
- `insertion point` before/after another element
- `line` by numeric index, as a range of elements, before/after another element
- `text` as a range of elements

Table 0.45 Text Class Properties

Name	Property Type
`offset` (read only)	`integer`
`length` (read only)	`integer`

Selection-Object

Table A.46 describes the Selection-Object class properties. The elements of this object are:

- `character` by numeric index, before/after another element, as a range of elements, or satisfying a test
- `line` by numeric index, as a range of elements, or before/after another element
- `text` as a range of elements

Table 0.46 Selection-Object Class Properties

Name	Property Type
`contents`	`type class`
`length` (read only)	`integer`
`offset` (read only)	`integer`

Syntax Coloring

Table A.48 describes the Syntax Coloring class properties.

Table 0.47 Syntax Coloring Class Properties

Name	Property Type
`Syntax coloring`	`boolean`
`Comment color`	`RGB color values list`
`Keyword color`	`RGB color values list`
`String color`	`RGB color values list`
`Custom color 1`	`RGB color values list`
`Custom color 2`	`RGB color values list`
`Custom color 3`	`RGB color values list`
`Custom color 4`	`RGB color values list`

Window

Table A.48 describes the Window class properties. The plural form of this object is Windows.

Table 0.48 Window Class Properties

Name	Property Type
`name`	`string`
`index`	`integer`
`bounds`	`bounding rectangle`
`document` (read only)	`document`
`position` (read only)	`point`
`visible` (read only)	`boolean`
`zoomed`	`boolean`

Object Classes

The object classes describe the properties of the objects in the project.

- Member Function Class
- Base Class
- Class Class
- Data Member Class

Member Function Class

Table A.49 lists the Member Function AppleScript class properties. The plural reference to use would be Member Functions.

Table 0.49 Member Function Class

Name	Property Type
`name` (read only)	`string`
`access` (read only)	• `public` • `pro-tected` • `private`
`virtual` (read only)	`boolean`
`static` (read only)	`boolean`
`declaration file` (read only)	`file specifi-cation`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`
`implementation file` (read only)	`file specifi-cation`

Name	Property Type
`implementation end offset` (read only)	`integer`
`implementation start offset` (read only)	`integer`

Base Class

Table A.50 lists the Base Class AppleScript class properties. The plural reference to use would be Base Classes.

Table 0.50 Base Class AppleScript Class Properties

Name	Property Type
`class` (read only)	`reference`
`access` (read only)	• `public` • `protected` • `private`
`virtual` (read only)	`boolean`

Class Class

Table A.51 lists the Class AppleScript class properties. The plural reference to use would be Classes. The elements of this class include:

- `base class` by numeric index
- `member function` by numeric index, and by name
- `data member` by numeric index, and by name

Table 0.51 Class AppleScript Class Properties

Name	**Property Type**
`name` (read only)	`string`
`language` (read only)	• `C` • `C++` • `Pascal` • `Object Pascal` • `Java` • `Assembler` • `Unknown`
`declaration file` (read only)	`file specification`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`
`subclasses` (read only)	`list of class`
`all subclasses` (read only)	`list of class`

Data Member Class

Table A.52 lists the Data Member AppleScript class properties. The plural reference to use would be Data Members.

Table 0.52 Data Member AppleScript Class Properties

Name	Property Type
`name` (read only)	`string`
`access` (read only)	• `public` • `private` • `protected`

Name	Property Type
`static` (read only)	`boolean`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`

Misc Classes

The Miscellaneous classes allow configuration of Version Control Systems, and Extras for the project settings.

- Extras
- Target
- Target File
- Text Document
- Version Control System Setup

Other classes are used for inheritance functions:

- Application
- Build Progress Document
- Catalog Document
- Class Browser
- Class Hierarchy
- Editor Document
- File
- File Compare Document
- Message Document
- Project Document
- Project Inspector
- Single Class Browser
- Single Class Hierarchy

- Symbol Browser
- ToolServer Worksheet

Extras

Table A.53 lists the Extras class properties.

Table 0.53 Extras Class

Name	Property Type
`Full screen zoom`	`boolean`
`External Reference`	• `Think Reference` • `QuickView`
`Use Script Menu`	`boolean`
`Use Editor Extensions`	`boolean`
`Use External Editor`	`boolean`

Target

Table A.54 lists the properties for the Target class. The plural form of Target is Targets. This class inherits all properties and elements of the given class.

Table 0.54 Target Class

Name	Property Type
`name`	`string`
`index` (read only)	`integer`
`project document` (read only)	`project document`

Target File

Table A.55 lists the properties for the Target File class. The plural form of Target File is Target Files.

Table 0.55 Target File Class

Name	Property Type
`id` (read only)	`integer`
`type` (read only)	• `library file` • `project file` • `resource file` • `text file` • `unknown file`
`index` (read only)	`integer`
`location` (read only)	`file specification`
`path` (read only)	`string`
`linked` (read only)	`boolean`
`link index` (read only)	`integer`
`modified date` (read only)	`date`
`compiled date` (read only)	`date`
`code size` (read only)	`integer`
`data size` (read only)	`integer`
`debug`	`boolean`
`weak link` (read only)	`boolean`
`init before`	`boolean`
`prerequisites` (read only)	`list of list`
`dependents` (read only)	`list`

Text Document

Table A.56 lists the properties for the Text Document class. The plural form of Text Document is Text Documents.

Table 0.56 Text Document Class

Name	Property Type
`inherits` (read only)	`document`
`modified` (read only)	`boolean`
`selection`	`selection-object`

Version Control System Setup

Table A.57 lists the VCS Setup class properties.

Table 0.57 Version Control System Class

Name	Property Type
`VCS Active`	`boolean`
`Connection Method`	`string`
`Username`	`string`
`Password`	`string`
`Auto Connect`	`boolean`
`Store Password`	`boolean`
`Always Prompt`	`boolean`
`Mount Volume`	`boolean`
`Database Path`	`path information`
`Local Root`	`path information`

Application

The Application class elements are:

- `document` by numeric index, by name, and as a range of elements
- `window` by numeric index, by name, and as a range of elements

Build Progress Document

The plural form of Build Progress Document is Build Progress Documents. This class inherits all properties and elements of the given class.

Catalog Document

The plural form of Catalog Document is Catalog Documents. This class inherits all properties and elements of the given class.

Class Browser

The plural form of Class Browser is Class Browsers. This class inherits all properties and elements of the given class.

Class Hierarchy

The plural form of Class Hierarchy is Class Hierarchies. This class inherits all properties and elements of the given class.

Editor Document

The plural form of Editor Document is Editor Documents. This class inherits all properties and elements of the given class.

File

The File Class plural to use in AppleScripts is Files.

File Compare Document

The plural form of File Compare Document is File Compare Documents. This class inherits all properties and elements of the given class.

Message Document

The plural form of Message Document is Message Documents. This class inherits all properties and elements of the given class.

Project Document

The plural form of Project Document is Project Documents. This class inherits all properties and elements of the given class.

Project Inspector

The plural form of Project Inspector is Project Inspectors. This class inherits all properties and elements of the given class.

Single Class Browser

The plural form of Single Class Browser is Single Class Browsers. This class inherits all properties and elements of the given class.

Single Class Hierarchy

The plural form of Single Class Hierarchy is Single Class Hierarchies. This class inherits all properties and elements of the given class.

Symbol Browser

The plural form of Symbol Browser is Symbol Browsers. This class inherits all properties and elements of the given class.

ToolServer Worksheet

The plural form of ToolServer Worksheet is ToolServer Worksheets. This class inherits all properties and elements of the given class.

Coding with CodeWarrior IDE and Apple Events

You may want to use low-level Apple Events instead of writing AppleScripts if you are producing tools or programs that need to control the CodeWarrior IDE while they are running. Third-party editors or browsers, and other tools, might require this capability.

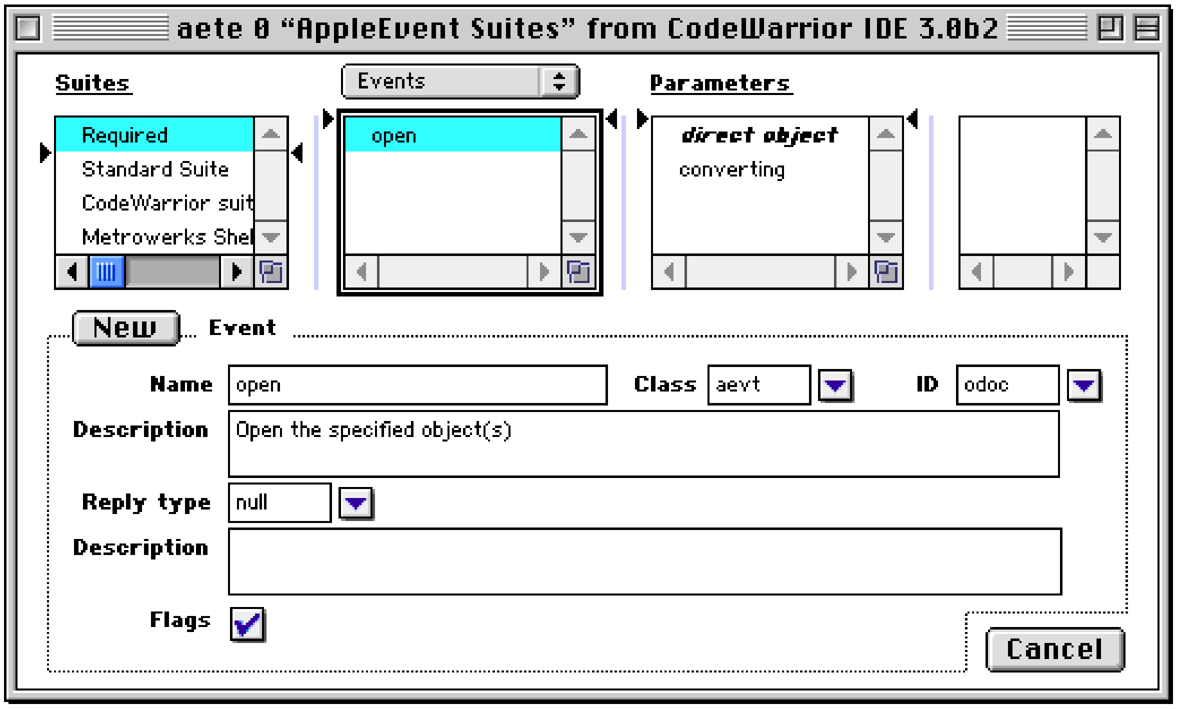
For documentation on using low-level Apple Events in your program code, refer to *Inside Macintosh: Interapplication Communication*

(Addison-Wesley) for a discussion of how to use the Apple Events portion of the Mac OS Toolbox.

There is some example code available that shows how to send Apple Events to the CodeWarrior IDE. You can find it on the CodeWarrior Reference CD in the `CodeWarrior Examples` folder, under the `MacOS Examples` folder. This code is a starter project for your work, and you will need to verify the code for proper operation. It is not intended to be a commercially-shipping product.

Largely, you will need to inspect the CodeWarrior IDE's '`aete`' and '`aedt`' resources using a resource editor to see what the low-level codes are to control the IDE. Figure A.1 shows an example view of what this might look like using the Resorcerer 2.0 resource editor. Rather than document all the low-level codes required to control the IDE, using a resource editor is the best solution to learn the low-level codes for now. With new innovations for the IDE on the horizon, the low-level codes may be documented at a later date.

Figure 0.1 Resorcerer 2.0 View of the CodeWarrior IDE 'aete' resource



Name: open
Class: aevt
ID: odoc
Description: Open the specified object(s)
Reply type: null
Description:
Flags: ✓

3

CodeWarrior Scripting on Microsoft Windows

This chapter introduces and discusses the scripting support provided by the CodeWarrior IDE on Microsoft Windows.

CodeWarrior Windows Scripting Overview

This chapter discusses the COM scripting classes supported in CodeWarrior and how to begin using them. You should read this chapter if you would like to enhance and extend the capabilities of the CodeWarrior IDE.

By scripting the IDE using a scripting editor, it is possible to execute many CodeWarrior IDE commands without using the IDE directly. Scripting the CodeWarrior IDE is a way to automate repetitive tasks that do not need user interaction.There are many exciting things that you can do to harness the power of the IDE, such as automate builds, generate files automatically, and configure settings.

TIP: Look at the example scripts on the CD. Reviewing these scripts will save you time when learning to write your own.

This chapter is not necessarily a tutorial. If you want to learn how to edit, save, and run scripts, you may not find the information here. Instead, refer to other tools and sources of information listed in “Tools and Reference Material” on page 86 for more information.

The topics in this chapter are:

Tools and Reference Material

There are several items you will want to become acquainted with in order to effectively script the IDE:

- Microsoft Scripting Technologies Web Site
- OLE/COM Object Viewer

Microsoft Scripting Technologies Web Site

You can find numerous resources and pointers to information on the world wide web at:

`http://msdn.microsoft.com/scripting/`

This address contains information about the Windows Scripting Host, the languages you can use for scripting, terminology, and debugging information.

OLE/COM Object Viewer

You will need a copy of the OLE/COM Object Viewer application from Microsoft in order to understand how to interpret the interfaces to the IDE. You can find this application on the world wide web at:

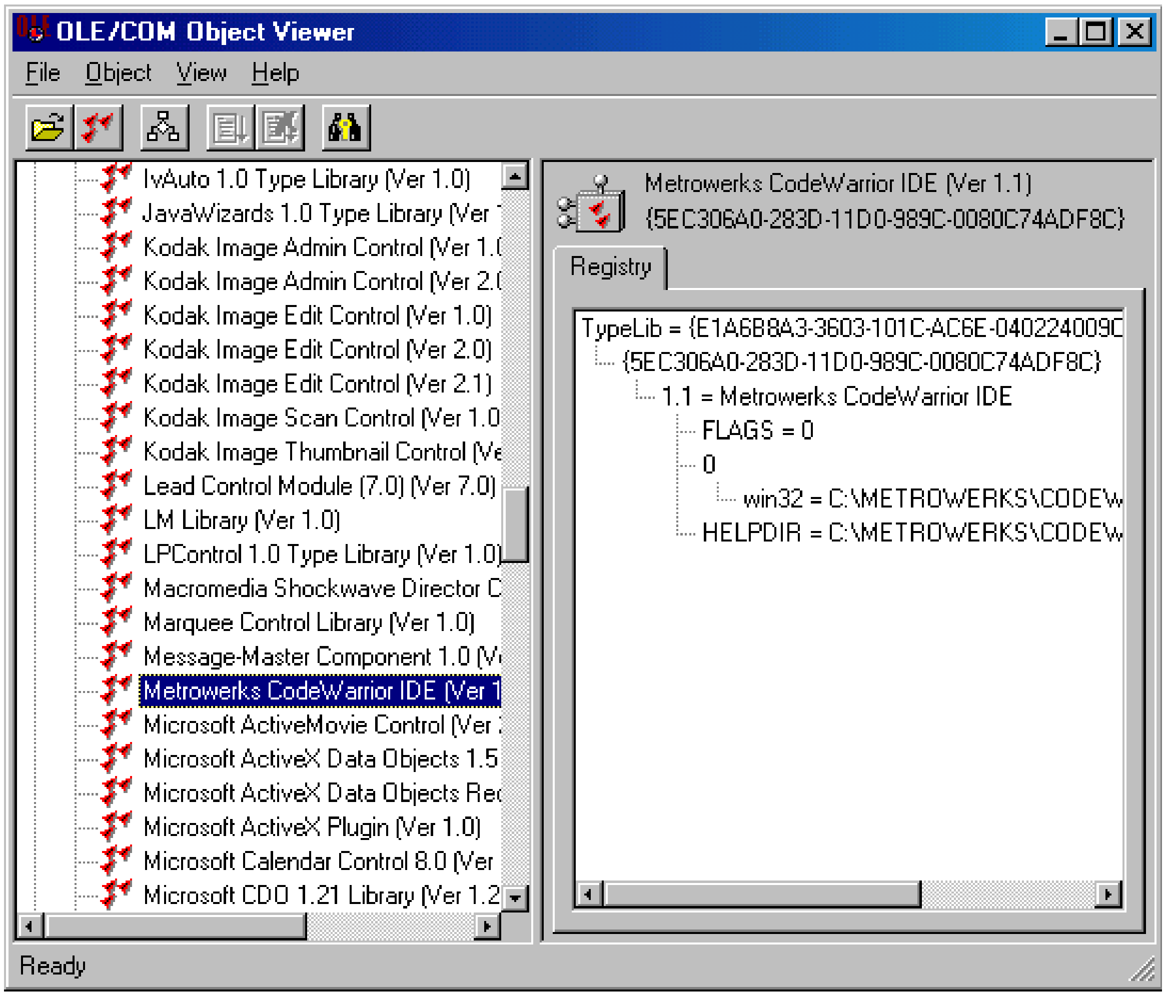
`http://www.microsoft.com/com/resources/oleview.asp`

Other editing and debugging tools may be available from third-party vendors.

To use OLE/COM Object Viewer, launch it and you will see something similar to Figure 3.1. Use the **View** menu to put it into **Expert Mode**, clickon the plus sign "+" next to **Type Libraries** in the left window pane, then click on the **Metrowerks CodeWarrior IDE** entry.

NOTE: In Visual Studio 6.0 tools suite from Microsoft, the application is instead called OLE Viewer.

Figure 3.1 OLE/COM Object Viewer



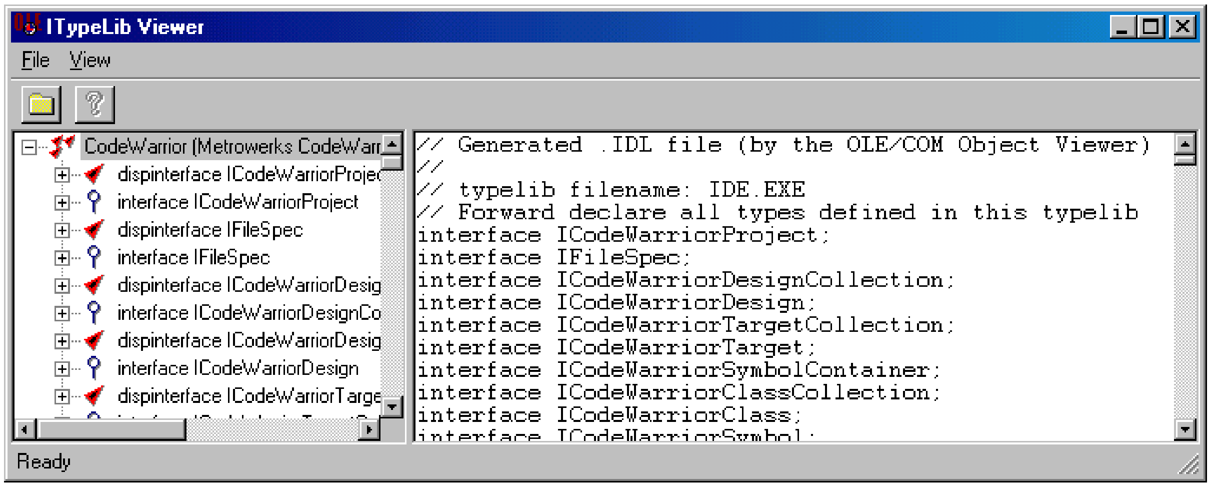
IvAuto 1.0 Type Library (Ver 1.0)
JavaWizards 1.0 Type Library (Ver 1
Kodak Image Admin Control (Ver 1.(
Kodak Image Admin Control (Ver 2.(
Kodak Image Edit Control (Ver 1.0)
Kodak Image Edit Control (Ver 2.0)
Kodak Image Edit Control (Ver 2.1)
Kodak Image Scan Control (Ver 1.0
Kodak Image Thumbnail Control (Ve
Lead Control Module (7.0) (Ver 7.0)
LM Library (Ver 1.0)
LPControl 1.0 Type Library (Ver 1.0)
Macromedia Shockwave Director C
Marquee Control Library (Ver 1.0)
Message-Master Component 1.0 (V
Metrowerks CodeWarrior IDE (Ver 1
Microsoft ActiveMovie Control (Ver
Microsoft ActiveX Data Objects 1.5
Microsoft ActiveX Data Objects Rec
Microsoft ActiveX Plugin (Ver 1.0)
Microsoft Calendar Control 8.0 (Ver
Microsoft CDO 1.21 Library (Ver 1.2

Metrowerks CodeWarrior IDE (Ver 1.1)
{5EC306A0-283D-11D0-989C-0080C74ADF8C}

```
TypeLib = {E1A6B8A3-3603-101C-AC6E-040224009C
  {5EC306A0-283D-11D0-989C-0080C74ADF8C}
    1.1 = Metrowerks CodeWarrior IDE
      FLAGS = 0
      0
        win32 = C:\METROWERKS\CODEW
      HELPDIR = C:\METROWERKS\CODEW
```

If you double-click on the **Metrowerks Codewarrior IDE** entry, a window like that shown in Fig appears.

Figure 3.2 ITypeLib Viewer


ITypeLib Viewer
File View
CodeWarrior (Metrowerks CodeWarr
dispinterface ICodeWarriorProjec
interface ICodeWarriorProject
dispinterface IFileSpec
interface IFileSpec
dispinterface ICodeWarriorDesig
interface ICodeWarriorDesignCo
dispinterface ICodeWarriorDesig
interface ICodeWarriorDesign
dispinterface ICodeWarriorTarge
Ready

```
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib filename: IDE.EXE
// Forward declare all types defined in this typelib
interface ICodeWarriorProject;
interface IFileSpec;
interface ICodeWarriorDesignCollection;
interface ICodeWarriorDesign;
interface ICodeWarriorTargetCollection;
interface ICodeWarriorTarget;
interface ICodeWarriorSymbolContainer;
interface ICodeWarriorClassCollection;
interface ICodeWarriorClass;
interface ICodeWarriorSymbol;
```

This view shows all the IDL (Interface Definition Language) for the COM object. Using this information you can determine how to talk to the IDE to control its behavior through scripting.

CodeWarrior IDE COM Classes

CodeWarrior events have several classes to let you control the CodeWarrior IDE's actions and option settings.

The following is a list of all interface types in the `.IDL` definition for the CodeWarrior IDE application. You can inspect the details of these definitions in the OLE/COM Object Viewer application.

Listing 3.1 IDL Types for the CodeWarrior IDE

```
interface ICodeWarriorProject;
interface IFileSpec;
interface ICodeWarriorDesignCollection;
interface ICodeWarriorDesign;
interface ICodeWarriorTargetCollection;
interface ICodeWarriorTarget;
interface ICodeWarriorSymbolContainer;
interface ICodeWarriorClassCollection;
interface ICodeWarriorClass;
interface ICodeWarriorSymbol;
interface ICodeWarriorSourceContext;
interface ICodeWarriorBaseClassCollection;
interface ICodeWarriorBaseClassInfo;
interface ICodeWarriorDataMemberCollection;
interface ICodeWarriorDataMember;
interface ICodeWarriorMethodCollection;
interface ICodeWarriorMethod;
interface ICodeWarriorProjectFileCollection;
interface ICodeWarriorProjectFile;
interface ICodeWarriorVCSState;
interface ICodeWarriorTargetFileCollection;
interface ICodeWarriorTargetFile;
interface ICodeWarriorAccessPaths;
interface ICodeWarriorAccessPathCollection;
interface ICodeWarriorAccessPath;
interface ICodeWarriorUserTree;
interface ICodeWarriorUserTreeCollection;
interface ICodeWarriorSubTargetCollection;
interface ICodeWarriorSubTarget;
interface IFileSpecCollection;
interface IBSTRCollection;
interface IStream;
interface ISequentialStream;
interface ICodeWarriorBuildMessages;
interface ICodeWarriorMessageCollection;
interface ICodeWarriorMessage;
interface ICodeWarriorTargetOutput;
interface ICodeWarriorApp;
interface ICodeWarriorProjectCollection;
interface ICodeWarriorCreatableItemCollection;
interface ICodeWarriorCreatableItem;
interface ICodeWarriorDocumentCollection;
interface ICodeWarriorDocument;
interface ICodeWarriorProjectDocument;
interface ICodeWarriorVersionControl;
interface ICodeWarriorTextDocument;
interface ICodeWarriorTextEngine;
interface ICodeWarriorComponent;
interface ICodeWarriorComponentPropertyCollection;
interface ICodeWarriorComponentProperty;
interface ICodeWarriorComponentEventSetCollection;
interface ICodeWarriorComponentEventSet;
interface ICodeWarriorComponentEventCollection;
interface ICodeWarriorComponentEvent;
interface ICodeWarriorSymbolCollection;
interface ICodeWarriorComponentCollection;
interface ICodeWarriorAppEvents;
interface ICodeWarriorProjectEvents;
interface ICodeWarriorDesignEvents;
interface ICodeWarriorDesignAttachment;
interface ICodeWarriorCreateProjectItem;
interface ICodeWarriorCreateFileItem;
interface ICodeWarriorCreateObjectItem;
interface ICodeWarriorVCSFileStateListener;
interface ICodeWarriorProjectAssociation;
interface ICodeWarriorErrorInfo;
```

Sample Scripts and How to Get Started

This section will show you how to get a jump on starting to write your own scripts.

The topics in this section are:

- How to Start Scripting
- Script Examples

How to Start Scripting

Note that one of the first lines of your script should be something like:

```
set codewarrior = CreateObject("CodeWarrior.CodeWarriorApp")
```

This creates a COM instance of the CodeWarrior IDE that you can interact with in subsequent scripting operations.

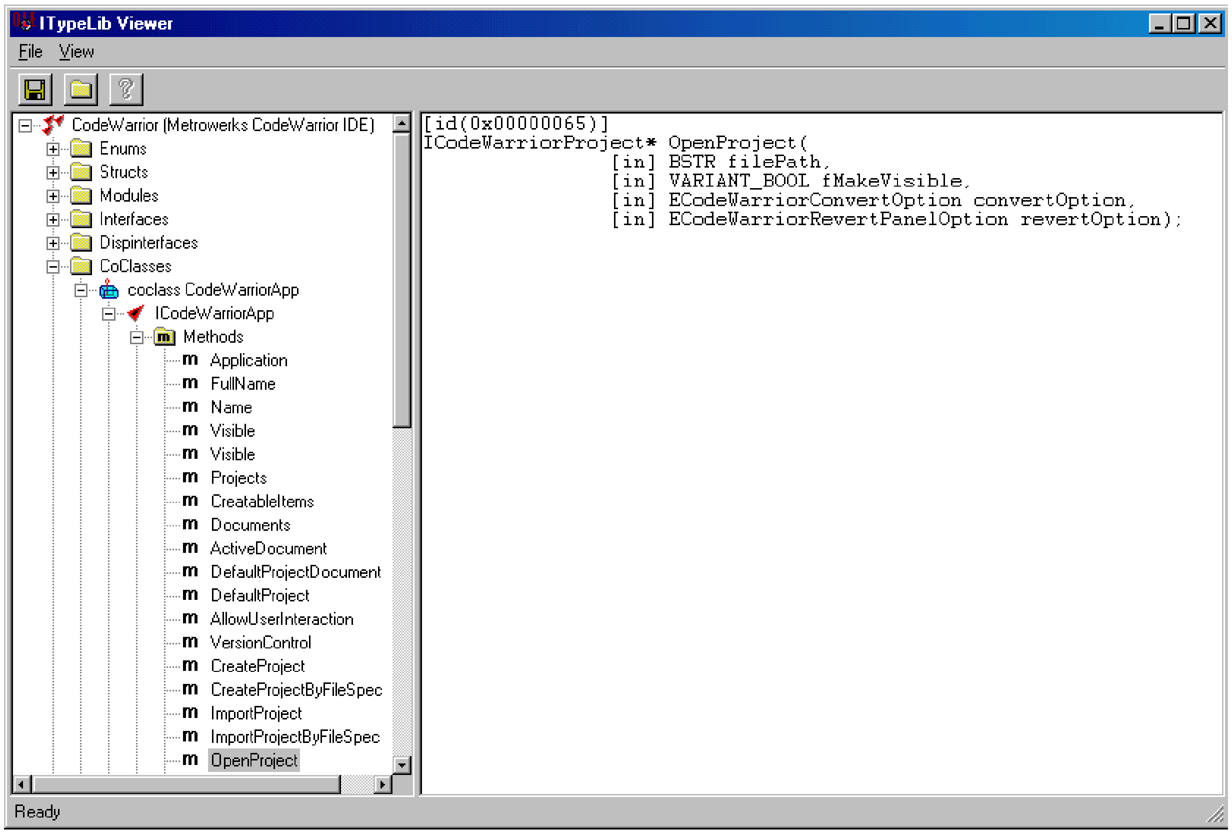
Now we'll analyze how you would use OLE/COM Object Viewer to learn how to script the IDE for one simple operation. In order to do much in the IDE, you will need to open a project. This can be accomplished with this script command:

```
set project = codewarrior.OpenProject(projectname, true, 2, 0 )
```

How would you know how to write this command? Simple, use the OLE/COM Object Viewer to inspect the CoClasses for the operation you want to perform.

To do this for the `OpenProject` method, you would first launch OLE/COM Object Viewer. Then use the **View** menu to put it into **Expert Mode**, clickon the plus sign "+" next to **Type Libraries** in the left window pane, then double-click on the **Metrowerks CodeWarrior IDE** entry. A window similar to Figure 3.2 on page 88 appears. Then use the **View** menu of the **ITypeLib Viewer** window to choose **Group by type kind**. Then click on the plus sign "+" next to **CoClasses**, then the + next to **coclass CodeWarriorApp**, then the + next to **Methods**. Click on **OpenProject** and your window should now look something like that shown in Figure 3.3.

Figure 3.3 OpenProject Method



```
[id(0x00000065)]
ICodeWarriorProject* OpenProject(
                [in] BSTR filePath,
                [in] VARIANT_BOOL fMakeVisible,
                [in] ECodeWarriorConvertOption convertOption,
                [in] ECodeWarriorRevertPanelOption revertOption);
```

This shows that the `OpenProject` method requires 4 arguments, all are parameters that are passed in only (the `[in]` designation signifies this):

- `filePath` - the path to the project to be opened.
- `fMakeVisible` - whether to make the project visible or not.
- `convertOption` - whether to allow the project to be converted or not (taking a value from the `ECodeWarriorConvertOption` enumeration).
- `revertOption` - the value from the enumeration named `ECodeWarriorRevertPanelOption` .

You can view the values for the enumerations in the same window under the **Enums** hierarchy.

Script Examples

Here are a couple of example scripts that illustrate the way you might want to write your scripts. These scripts are invoked from the command-line (**Start/Programs/MS-DOS Prompt**) using the `wscript` executable.

The scripts in this section are:

- Removing Object Code
- Build and Wait

Removing Object Code

This script accepts as the command-line parameter the path to the project to be opened. The absolute path needs to be included, for example:

```
wscript select~1.vbs "C:\testproejcts\test1.mcp”
```

If no command line arguments are given, the script prompts the user for the absolute path of the project file to be opened. If specified, the script tries to open the project, otherwise it opens the default one c:\testprojects\test1.mcp. This script opens the project and selects the files that belong to the default target.

Listing 3.2 RemoveObjectCode.vbs

```
option explicit

'*******Variable declaration
dim codewarrior
dim project
dim projectname
dim targetIntf
dim count
dim projectCollection
dim targetcollection
dim result
dim showinputbox
dim objArgs
```

```
'****** Script ********
Set objArgs = Wscript.Arguments
projectname = "c:\testprojects\test1.mcp"

if objArgs.Count > 1 then
  MsgBox "This Script expects only one argument, rest of the
arguments will be ignored!!"
  showinputbox = false
  projectname = CStr(objArgs(0))
end if

if objArgs.Count = 0 then
  showinputbox = true
else
  showinputbox = false
  projectname = CStr(objArgs(0))
end if

if showinputbox = true then
  result = InputBox("Enter the absolute path for the project to be
opened","Input", projectname, 100, 100)

  If result = "" Then
    projectname = "c:\testprojects\test1.mcp"
  else
    projectname = cstr(result)
  end if
end if

'Create automation app object
set codewarrior = CreateObject("CodeWarrior.CodeWarriorApp")
MsgBox "App Created"

project = Null
'open project
set project = codewarrior.OpenProject(projectname, true, 2, 0 )
if TypeName( project ) <> "Null"  then
  set targetcollection = project.Targets
  count = targetcollection.Count
  IF  ( count > 0 ) then
    set targetIntf = targetcollection.Item( 0 )
    targetIntf.RemoveObjectCode( true )
  END IF
else
  MsgBox CStr( projectname & " does not exist" )
end if
```

Build and Wait

This script accepts as commandline parameter the name of the project to be opened. The absolute path needs to be included, for example:

```
wscript select~1.vbs "C:\testprojects\test1.mcp"
```

If no command-line arguments are given, the script prompts the user for the absolute path of the project file to be opened. If specified, the script tries to open the project, else opens te default one "c:\testprojects\test1.mcp”.

Listing 3.3 BuildAndWait.vbs

```
option explicit

'*******Variable declaration
dim codewarrior
dim project
dim projectname
dim targetIntf
dim count
dim projectCollection
dim targetcollection
dim result
dim showinputbox
dim objArgs
dim buildErrors
```

```
'****** Script ********
Set objArgs = Wscript.Arguments
projectname = "c:\temp\none\none.mcp"

if objArgs.Count > 1 then
  MsgBox "This Script expects only one argument, rest of the
arguments will be ignored!!"
  showinputbox = false
  projectname = CStr(objArgs(0))
end if

if objArgs.Count = 0 then
  showinputbox = true
else
  showinputbox = false
  projectname = CStr(objArgs(0))
end if

if showinputbox = true then
  result = InputBox("Enter the absolute path for the project to be
opened","Input", projectname, 100, 100)

  If result = "" Then
    projectname = "c:\testprojects\test1.mcp"
  else
    projectname = cstr(result)
  end if
end if

'Create automation app object
set codewarrior = CreateObject("CodeWarrior.CodeWarriorApp")
MsgBox "App Created"

project = Null
'open project
set project = codewarrior.OpenProject(projectname, true, 2, 0 )
if TypeName( project ) <> "Null"  then
  project.BuildAndWaitToComplete
else
  MsgBox CStr( projectname & " does not exist" )
```

```
end if

project.close
```

Index

CodeWarrior

Scripting Reference

Credits

writing lead:	BitHead
other writers:	Carl B. Constantine and Stephen Chernicoff, Alisa Dean
engineering:	Berardino Barratta, Dan Podwall, Fred Peterson, Benjamin Koznik, Marcel Achim, Glenn Meter, Lawrence You
frontline warriors:	Richard Atwell, John Daub, Patrick Sullivan, Derek Saldana, David Blache, Kevin Bell, Matt Henderson, Ron Liechty, Metrowerks Support, and CodeWarrior users everywhere

metrowerks®

Guide to CodeWarrior Documentation

CodeWarrior documentation is modular, like the underlying tools. There are manuals for the core tools, languages, libraries, and targets. The exact documentation provided with any CodeWarrior product is tailored to the tools included with the product. Your product will not have every manual listed here. However, you will probably have additional manuals (not listed here) for utilities or other software specific to your product.

Core Documentation	
IDE User Guide	How to use the CodeWarrior IDE
Debugger User Guide	How to use the CodeWarrior debugger
CodeWarrior Core Tutorials	Step-by-step introduction to IDE components
Language/Compiler Documentation	
C Compilers Reference	Information on the C/C++ front-end compiler
Pascal Compilers Reference	Information on the Pascal front-end compiler
Error Reference	Comprehensive list of compiler/linker error messages, with many fixes
Pascal Language Reference	The Metrowerks implementation of ANS Pascal
Assembler Guide	Stand-alone assembler syntax
Command-Line Tools Reference	Command-line options for Mac OS and Be compilers
Plugin API Manual	The CodeWarrior plugin compiler/linker API
Library Documentation	
MSL C Reference	Function reference for the Metrowerks ANSI standard C library
MSL C++ Reference	Function reference for the Metrowerks ANSI standard C++ library
Pascal Library Reference	Function reference for the Metrowerks ANS Pascal library
MFC Reference	Reference for the Microsoft Foundation Classes for Win32
Win32 SDK Reference	Microsoft’s Reference for the Win32 API
The PowerPlant Book	Introductory guide to the Metrowerks application framework for Mac OS
PowerPlant Advanced Topics	Advanced topics in PowerPlant programming for Mac OS
Targeting Manuals	
Targeting BeOS	How to use CodeWarrior to program for BeOS
Targeting Java VM	How to use CodeWarrior to program for the Java Virtual Machine
Targeting Mac OS	How to use CodeWarrior to program for Mac OS
Targeting MIPS	How to use CodeWarrior to program for MIPS embedded processors
Targeting NEC V810/830	How to use CodeWarrior to program for NEC V810/830 processors
Targeting Net Yaroze	How to use CodeWarrior to program for Net Yaroze game console
Targeting Nucleus	How to use CodeWarrior to program for the Nucleus RTOS
Targeting OS-9	How to use CodeWarrior to program for the OS-9 RTOS
Targeting Palm OS	How to use CodeWarrior to program for PalmPilot
Targeting PlayStation OS	How to use CodeWarrior to program for the PlayStation game console
Targeting PowerPC Embedded Systems	How to use CodeWarrior to program for PPC embedded processors
Targeting VxWorks	How to use CodeWarrior to program for the VxWorks RTOS
Targeting Win32	How to use CodeWarrior to program for Windows