



# **CodeWarrior™ Development Tools COM API Reference**

Revised: 2003/08/19



Freescale, the Freescale insignia, and CodeWarrior are registered trademarks of Freescale Corp. in the US and/or other countries. All other trade names, trademarks and registered trademarks are the property of their respective owners.

© Copyright 2006. Freescale Corp. ALL RIGHTS RESERVED.

Freescale reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Freescale does not assume any liability arising out of the application or use of any product described herein. Freescale software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft or automobile navigation or communication, emergency systems, or other applications with a similar degree of potential hazard.

Documentation stored on electronic media may be printed for personal use only. Except for the forgoing, no portion of this documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, without prior written permission from Freescale.

ALL SOFTWARE, DOCUMENTATION AND RELATED MATERIALS ARE SUBJECT TO THE FREESCALE END USER LICENSE AGREEMENT FOR SUCH PRODUCT.

## How to Contact Freescale:

<b>Corporate Headquarters</b>	Freescale Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
<b>World Wide Web</b>	<a href="http://www.freescale.com">http://www.freescale.com</a>
<b>Ordering &amp; Technical Support</b>	Voice: (800) 377-5416 Fax: (512) 996-4910

# Table of Contents

---

<b>1 Introduction</b>	<b>9</b>
Overview . . . . .	9
Read the Release Notes! . . . . .	10
About This Manual . . . . .	10
Requirements . . . . .	11
What You Should Already Know . . . . .	11
Starting Points . . . . .	11
Services . . . . .	11
Callbacks . . . . .	12
 <b>2 Access Paths</b>	 <b>13</b>
Access Paths API Reference . . . . .	13
IcodeWarriorAccessPath . . . . .	14
IcodeWarriorAccessPaths . . . . .	19
IcodeWarriorUserTree . . . . .	26
Access Paths Data Types . . . . .	30
 <b>3 Application</b>	 <b>33</b>
Application API Overview . . . . .	33
Application API Reference . . . . .	34
IcodeWarriorApp . . . . .	35
IcodeWarriorAppEvents . . . . .	65
IcodeWarriorCompare . . . . .	69
Application Data Types . . . . .	72
 <b>4 Collections</b>	 <b>77</b>
Collections API Overview . . . . .	77
Using the Collections API . . . . .	77
Collections API Reference . . . . .	78
CodeWarrior Collections . . . . .	79
 <b>5 Commands</b>	 <b>83</b>
Commands API Overview . . . . .	83
Using the Commands API . . . . .	84
Commands API Reference . . . . .	88
IcodeWarriorCommandHandler . . . . .	89

IcodeWarriorCommandRegistry . . . . .	92
IcodeWarriorDeferredAction . . . . .	94
Commands Data Types . . . . .	95
<b>6 Components</b>	<b>99</b>
Components API Overview . . . . .	99
Components API Reference . . . . .	99
IcodeWarriorComponent . . . . .	100
IcodeWarriorComponentEvent . . . . .	104
IcodeWarriorComponentEventSet . . . . .	107
IcodeWarriorComponentProperty . . . . .	109
<b>7 Creatable Items</b>	<b>113</b>
Creatable Items API Reference . . . . .	113
IcodeWarriorCreatableItem . . . . .	114
IcodeWarriorCreateFileItem . . . . .	117
IcodeWarriorCreateObjectItem . . . . .	120
IcodeWarriorCreateProjectItem . . . . .	124
Creatable Items Data Types . . . . .	127
<b>8 Designs</b>	<b>129</b>
Designs API Overview . . . . .	129
Designs API Reference . . . . .	129
IcodeWarriorDesign . . . . .	130
IcodeWarriorDesignAttachment . . . . .	141
IcodeWarriorDesignEvents . . . . .	143
Data Types . . . . .	145
<b>9 Dialog Services</b>	<b>147</b>
Dialog Services API Overview . . . . .	147
Using the Dialog Services API . . . . .	147
Registering the Command . . . . .	147
Implementing the Command . . . . .	149
Dialog Services API Reference . . . . .	151
IcodeWarriorDialogServices . . . . .	151
Dialog Services Data Types . . . . .	157

---

<b>10 Documents</b>	<b>159</b>
Documents API Overview . . . . .	159
Documents API Reference . . . . .	159
IcodeWarriorDocument. . . . .	160
IcodeWarriorProjectDocument. . . . .	168
IcodeWarriorTextDocument. . . . .	171
<b>11 Error Info</b>	<b>175</b>
Error Info API Overview. . . . .	175
Error Info API Reference. . . . .	175
IcodeWarriorErrorInfo . . . . .	176
<b>12 Files</b>	<b>185</b>
IfileSpec . . . . .	185
<b>13 Menus</b>	<b>189</b>
Menus API Overview . . . . .	189
Using the Menus API . . . . .	189
Menus API Reference . . . . .	190
IcodeWarriorMenu. . . . .	191
IcodeWarriorMenuHandler . . . . .	196
IcodeWarriorMenuManager. . . . .	198
<b>14 Messages</b>	<b>201</b>
Messages API Overview. . . . .	201
Messages API Reference. . . . .	201
IcodeWarriorBuildMessages. . . . .	202
IcodeWarriorMessage . . . . .	207
Message Data Types . . . . .	214
<b>15 Projects</b>	<b>215</b>
Projects API Overview . . . . .	215
Projects API Reference. . . . .	215
IcodeWarriorProject . . . . .	216
IcodeWarriorProjectAssociation . . . . .	237
IcodeWarriorProjectEvents . . . . .	239
IcodeWarriorProjectFile. . . . .	245
Project Data Types . . . . .	248

<b>16 Symbols</b>	<b>251</b>
Symbols API Reference . . . . .	251
IcodeWarriorBaseClassInfo . . . . .	252
IcodeWarriorClass . . . . .	254
IcodeWarriorDataMember . . . . .	261
IcodeWarriorMethod . . . . .	264
IcodeWarriorSourceContext . . . . .	270
IcodeWarriorSymbol . . . . .	274
IcodeWarriorSymbolContainer . . . . .	278
Symbols Data Types . . . . .	283
<b>17 Targets</b>	<b>285</b>
Targets API Overview . . . . .	285
Targets API Reference . . . . .	285
IcodeWarriorTarget . . . . .	286
IcodeWarriorTargetFile . . . . .	317
IcodeWarriorTargetOutput . . . . .	325
IcodeWarriorSubTarget . . . . .	327
IcodeWarriorSubProjectTarget . . . . .	329
Targets Data Types . . . . .	331
<b>18 Text</b>	<b>333</b>
Text API Overview . . . . .	333
Text API Reference . . . . .	333
IcodeWarriorTextEngine . . . . .	334
<b>19 Toolbar</b>	<b>343</b>
Toolbar API Overview . . . . .	343
Toolbar API Reference . . . . .	343
IcodeWarriorCustomToolbarItem . . . . .	344
IcodeWarriorPopupMenuToolbarItem . . . . .	347
IcodeWarriorToggleButtonToolbarItem . . . . .	352
IcodeWarriorToolbar . . . . .	354
IcodeWarriorToolbarInstanceCreationNotification . . . . .	361
IcodeWarriorToolbarItemHelp . . . . .	363
IcodeWarriorToolbarItemRegistry . . . . .	364
Toolbar Data Types . . . . .	366
Toolbar Constants . . . . .	367

---

<b>20 Version Control</b>	<b>369</b>
Version Control API Reference . . . . .	369
ICodeWarriorVersionControl . . . . .	370
ICodeWarriorVCSState . . . . .	375
ICodeWarriorVCSFileStateListener . . . . .	377
VCS Data Types . . . . .	378
<b>21 Windows</b>	<b>381</b>
Windows API Overview . . . . .	381
Using the Windows API . . . . .	381
Windows API Reference . . . . .	382
ICodeWarriorWindowManager . . . . .	383
ICodeWarriorWindow . . . . .	387
ICodeWarriorWindowEvents . . . . .	399
Windows Data Types . . . . .	406
<b>A CodeWarrior IDE Interface Definition Language (IDL)</b>	<b>407</b>
<b>Index</b>	<b>463</b>



## Table of Contents

---



# Introduction

---

Welcome to the CodeWarrior *COM API Reference*.

COM is Microsoft's Common Object Model. You can learn more about COM on the web at:

---

<http://msdn.microsoft.com>

---

For information on writing IDE plug-ins, see the *CodeWarrior IDE Plug-in Manual*. The information needed to write plug-ins is mainly in that manual.

This chapter contains the following sections:

- [Overview](#) — an introduction to this manual
- [Read the Release Notes!](#) — getting last minute information about creating plug-ins for the CodeWarrior IDE
- [Requirements](#) — what you'll need to develop plug-ins
- [What You Should Already Know](#) — what this manual assumes you know about using computers and computer programming
- [Starting Points](#) — how to use this manual

## Overview

This manual shows you how to use the COM API's to control CodeWarrior and modify interface elements and project-related data within the IDE.

The APIs are available to C++ and VB Script applications. The reference section of each chapter in this manual lists each method as it appears in each of these languages.

## Read the Release Notes!

Before referring to the rest of this manual, read the release notes! They contain important information about new features, bug fixes, and any late breaking changes.

## About This Manual

This manual uses some style conventions to make it easier to read and find specific information:

### Notes, warnings, tips, and beginner's hints

An advisory statement or **NOTE** may restate an important fact, or call your attention to a fact which may not be obvious.

A **WARNING** given in the text may call attention to something such as an irreversible operation or a possible error that may occur.

A **TIP** can help you become more productive with the CodeWarrior IDE.

A **For Beginners** note may help you better understand the terminology or concepts if you are new to programming.

### Typeface conventions

If you see some text that appears in a `different` typeface, you are reading file or folder names, source code, keyboard input, or programming items.

Text **formatted like this** means that the text refers to an item on the screen, such as a **menu command** or **control** in a dialog box.

If you are using an on-line viewing application that supports hypertext navigation, such as Adobe Acrobat, you can click on underlined and colored text to view another topic or related information.

## Requirements

To write COM programs that control the CodeWarrior IDE, you'll need a CodeWarrior package that comes with the tools and files needed to develop software for the operating system or computer platform on which your program will run.

Follow the instructions in the QuickStart guide of your CodeWarrior product to install the software.

## What You Should Already Know

The manual shows you how to use the CodeWarrior COM APIs in your programs to manipulate the CodeWarrior IDE.

It is assumed that you have some experience with an object-oriented language such as C++, Java, or VBScript and are familiar with Microsoft COM.

If you are not yet familiar with Microsoft COM, we recommend that you read *Inside COM - Microsoft's Common Object Model*, by Dale Rogerson. The following website has everything you need to learn the basics of COM, including tutorials:

<http://microsoft.com/com/default.asp>

## Starting Points

Fundamentally, there are two kinds of interfaces, Services and Callbacks. You can't tell which is which by looking at them.

### Services

Services are functions hidden inside the IDE that you can call for your purposes. All you need is the interface pointer. This includes the large majority of all defined interfaces.

Examples of services include:

ICodeWarriorApp

ICodeWarriorMenu

ICodeWarriorWindow

and many, many more.

## Callbacks

The IDE calls back into your code. A callback is a method that the IDE will call. It is probably not implemented inside the IDE. For such an interface, you inherit (subclass) and implement the abstract interface in full, since every method is pure virtual. You must write the whole thing from scratch. No inherited implementation, only inherited interface. In this way, COM is like multiple inheritance in Java, in which you inherit only interfaces.

Examples of callbacks include:

ICodeWarriorToggleButtonToolBarItem

ICodeWarriorPopupMenuToolBarItem

ICodeWarriorCustomToolBarItem

ICodeWarriorWindowEvents

ICodeWarriorCommandHandler

ICodeWarriorAppEvents

# Access Paths

---

This chapter shows how to use the Access Paths API to create and manipulate access paths in the CodeWarrior IDE.

## Access Paths API Reference

This section describes the methods contained in the following interfaces:

- [ICodeWarriorAccessPath](#)
- [ICodeWarriorAccessPaths](#)
- [ICodeWarriorUserTree](#)

The following data types are used with these interfaces:

- [EAccessPathLocation](#)
- [EUserDefinedTree](#)
- [EAccessPathType](#)

## ICodeWarriorAccessPath

This interface represents a CodeWarrior access path.

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface exposes the following methods:

<a href="#">get_AccessPathLocation</a>	<a href="#">get_SubDirectories</a>
<a href="#">get_AccessPathType</a>	<a href="#">get_UserTree</a>
<a href="#">get_Path</a>	<a href="#">put_AccessPathLocation</a>
<a href="#">get_Recursive</a>	<a href="#">put_Recursive</a>

---

### get\_AccessPathLocation

This method gets the origin of this access path.

```
virtual HRESULT AccessPathLocation(
    EAccessPathLocation *pval) = 0;
```

pval

On return, this parameter contains a pointer to a value within the range defined by the EAccessPathLocation enumeration, indicating the origin of this access path.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_AccessPathType

This method gets the type of this access path.

```
virtual HRESULT get_AccessPathType(  
    EAccessPathType *pval) = 0;
```

pval

On return, this parameter a pointer to a value in the range defined by the EAccessPathType enumeration, representing whether this access path is a user path or a system path.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Path

This method gets a file specification for this access path.

```
virtual HRESULT get_Path(IFileSpec **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the folder for this access path.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## Access Paths

### *get\_Recursive*

---

#### get\_Recursive

This method gets whether an access path is recursive or not.

```
virtual HRESULT get_Recursive(
    VARIANT_BOOL *pval) = 0;
```

pval

On return, this parameter contains a pointer to a boolean value that is set to `true` if this access path is recursive or `false` if this access path is not recursive.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

#### get\_SubDirectories

This method gets a list of all subfolders contained by the folder pointed to by this access path.

```
virtual HRESULT get_SubDirectories(
    ICodeWarriorAccessPathCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the list of access paths, one for each subfolder contained by the folder pointed to by this access path.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---



---

## get\_UserTree

If this access path has a user-defined origin, this method gets the corresponding user tree object. If this access path instead uses one of the origins defined in `EAccessPathLocation`, this method gets nothing.

```
virtual HRESULT get_UserTree(  
    ICodeWarriorUserTree **pval) = 0;
```

`pval`

On return, this parameter contains the address of a pointer to the user tree for this access path.

**Returns** `S_OK` if this method call succeeded or an appropriate error if it failed. This method can also return nothing if you specify one of the access paths defined in the `EAccessPathLocation` enumeration.

**See Also** [“EAccessPathLocation” on page 30](#)

---

## put\_AccessPathLocation

This method sets the origin of this access path.

```
virtual HRESULT AccessPathLocation(  
    EAccessPathLocation val) = 0;
```

`val`

Set this parameter to a value within the range defined by the `EAccessPathLocation` enumeration, indicating the origin of this access path.

**Returns** `S_OK` if this method call succeeded or an appropriate error if it failed.

## Access Paths

### *put\_Recursive*

---

## put\_Recursive

This method sets whether this access path is recursive.

```
virtual HRESULT put_Recursive(  
    VARIANT_BOOL pval) = 0;
```

pval

Set this parameter to `true` if this access path is recursive or `false` if this access path is not recursive.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorAccessPaths

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">ApplyChanges</a>	<a href="#">get_AlwaysSearchUserPaths</a>
<a href="#">CreateAccessPath</a>	<a href="#">get_SystemAccessPaths</a>
<a href="#">CreateAccessPathByFileSpec</a>	<a href="#">get_UserAccessPaths</a>
<a href="#">CreateAccessPathByPosition</a>	<a href="#">put_AlwaysSearchUserPaths</a>

### ApplyChanges

This method applies any changes you have made to this access path. You must call this method in order for changes you make to take effect.

```
virtual HRESULT ApplyChanges(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

### CreateAccessPath

This method creates a new access path by using a string to specify the new access path.

```
virtual HRESULT CreateAccessPath(
    BSTR path,
    VARIANT_BOOL Recursion,
    EAccessPathLocation inLocation,
    EAccessPathType inType,
```

## Access Paths

### CreateAccessPathByFileSpec

---

```
ICodeWarriorAccessPath **pval) = 0;
```

path

The full path to the folder you want to add.

Recursion

Set this parameter to `true` if you would like the CodeWarrior IDE to search subfolders of this access path. Otherwise, set it to `false`.

inLocation

A value within the range defined by the `EAccessPathLocation` enumeration, indicating the origin of the new access path.

inType

A value in the range defined by the `EAccessPathType` enumeration, indicating whether this is a user path or a system path.

pval

On return, this parameter contains the address of a pointer to the newly created access path.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## CreateAccessPathByFileSpec

This method creates a new access path by using the `IFileSpec` interface.

```
virtual HRESULT CreateAccessPathByFileSpec(
    IFileSpec *path,
    VARIANT_BOOL Recursion,
    EAccessPathLocation inLocation,
    EAccessPathType inType,
    ICodeWarriorAccessPath **pval) = 0;
```

`path`

A pointer to the `IFileSpec` interface representing the folder you want to add.

`Recursion`

Set this parameter to `true` if you would like the CodeWarrior IDE to search subfolders of this access path. Otherwise, set it to `false`.

`inLocation`

A value within the range defined by the `EAccessPathLocation` enumeration, indicating the origin of this access path.

`inType`

A value in the range defined by the `EAccessPathType` enumeration, indicating whether this is a user path or a system path.

`pval`

On return, this parameter contains the address of a pointer to the newly created access path.

**Returns** `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## CreateAccessPathByPosition

This method creates a new access path by using the `EAccessPathLocation` interface.

```
virtual HRESULT CreateAccessPathByPosition(  
    BSTR path,  
    VARIANT_BOOL Recursion,  
    EAccessPathLocation inLocation,  
    EAccessPathType inType,  
    BSTR userTreeName,  
    long position,  
    ICodeWarriorAccessPath **pval) = 0;
```

## Access Paths

### `get_AlwaysSearchUserPaths`

---

`path`

The path you want to add.

`Recursion`

Set this parameter to `true` if you would like the CodeWarrior IDE to search subfolders of this access path. Otherwise, set it to `false`.

`inLocation`

A value within the range defined by the `EAccessPathLocation` enumeration, indicating the origin of this access path.

`inType`

A value in the range defined by the `EAccessPathType` enumeration, indicating whether this is a user path or a system path.

`userTreeName`

The name of the user tree to which the path is being added.

`position`

An integer indicating the position for the new path.

`pval`

On return, this parameter contains the address of a pointer to the newaccess path.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## `get_AlwaysSearchUserPaths`

This method obtains the state of the **Always Search User Paths** option. When enabled, this option tells CodeWarrior to always search user paths before searching system paths.

```
virtual HRESULT get_AlwaysSearchUserPaths(
    VARIANT_BOOL *pval) = 0;
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if this option is enabled or `false` if this option is disabled.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## get\_SystemAccessPaths

This method gets a list of all system access paths in this collection. The system paths collection contains all compiler-relative paths.

```
virtual HRESULT get_SystemAccessPaths(  
    ICodeWarriorAccessPathCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all system access paths.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_UserAccessPaths

This method gets a list of all user access paths in this collection. The user paths collection contains all project-relative paths.

```
virtual HRESULT get_UserAccessPaths(  
    ICodeWarriorAccessPathCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all user access paths.



## Access Paths

*get\_UserAccessPaths*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)



---

## put\_AlwaysSearchUserPaths

This method sets the state of the **Always Search User Paths** option. When enabled, this option tells the CodeWarrior IDE to always search user paths before searching system paths.

```
virtual HRESULT put_AlwaysSearchUserPaths(  
    VARIANT_BOOL pval) = 0;
```

pval

Set this parameter to `true` if this option is enabled or `false` if this option is disabled.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorUserTree

The ICodeWarriorUserTree interface lets you work with user-defined access paths.

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">get_KeyName</a>	<a href="#">put_KeyName</a>
<a href="#">get_Name</a>	<a href="#">put_Name</a>
<a href="#">get_Type</a>	<a href="#">put_Type</a>
<a href="#">get_Value</a>	<a href="#">put_Value</a>

---

### get\_KeyName

This method gets the key name of the current user tree.

```
virtual HRESULT get_KeyName(
    BSTR *pval) = 0;
```

pval

On return, this parameter contains a pointer to the key name of the current user tree.

## get\_Name

This method gets the name of this user tree.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of this user tree.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Type

This method gets the type of this user tree.

```
virtual HRESULT get_Type(  
    EUserDefinedTree *val) = 0;
```

val

On return, this parameter contains a pointer to a value in the range defined by the EUserDefinedTree enumeration.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Value

This method gets the value of this user tree.

```
virtual HRESULT get_Value(  
    BSTR *pval) = 0;
```

## Access Paths

*put\_KeyName*

---

pval

On return, this parameter contains the value of this user tree.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_KeyName

This method sets the key name of the current user tree.

```
virtual HRESULT put_KeyName(
    BSTR val) = 0;
```

val

The string to which to set the key name of the current user tree.

---

## put\_Name

This method sets the name of this user tree.

```
virtual HRESULT put_Name(
    BSTR val) = 0;
```

val

The name of this user tree.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Type

This method sets the type of this user tree.

```
virtual HRESULT put_Type(
```

---

```
EUserDefinedTree val) = 0;
```

val

A value in the range defined by the EUserDefinedTree enumeration.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## put\_Value

This method sets the value of this user tree.

```
virtual HRESULT put_Value(  
    BSTR val) = 0;
```

val

The value of this user tree.

## Access Paths Data Types

The following data types are used with the Access Paths API:

- [EUserDefinedTree](#)
- [EAccessPathLocation](#)
- [EAccessPathType](#)

### EUserDefinedTree

This enumeration describes the type of a user tree.

**Table 2.1 EUserDefinedTree Enumeration**

Constant	Description
kAbsolutePath	An absolute (that is, fully qualified) file path.
kEnvironment	A path stored as an environment variable.
kRegistry	A path stored in the registry.

### EAccessPathLocation

This enumeration describes the origin of an access path.

**Table 2.2 EAccessPathLocation Enumeration**

Constant	Description
kAbsolute	An absolute (that is, fully qualified) file path.
kProjectRelative	A file path relative to the location of the project file.
kCompilerRelative	A file path relative to the location of the compiler's executable file.
kSystemRelative	A file path relative to the location of the operating system files.
kUserDefined	A file path relative to a user-defined location.

## EAccessPathType

This enumeration describes the type of an access path.

**Table 2.3 EAccessPathType Enumeration**

Constant	Description
kUserPath	A user-specified path.
kSystemPath	A system path.



## **Access Paths**

*Access Paths Data Types*

---



# Application

---

This chapter describes the Application API to work with and receive events from the CodeWarrior IDE about the application object..

This chapter contains the following sections:

- [Application API Overview](#)
- [Application API Reference](#)

## Application API Overview

The Application API is a set of interfaces that allows a plug-in to manipulate and receive events from the CodeWarrior IDE.

You can use the application object API to manipulate application properties, for project and target management, document and file management, command management, and other tasks.

## Application API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorApp](#)
- [ICodeWarriorAppEvents](#)
- [ICodeWarriorCompare](#)

These interfaces use various data types, which are described in the following section:

- [Application Data Types](#)

## ICodeWarriorApp

This is the CodeWarrior application object. Use it to manipulate application properties, for project and target management, document and file management, command management, and other miscellaneous tasks.

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">AddCreatableItem</a>	<a href="#">get_VersionControl</a>
<a href="#">AddUserTree</a>	<a href="#">get_Visible</a>
<a href="#">AttemptModify</a>	<a href="#">ImportProject</a>
<a href="#">CreateProject</a>	<a href="#">ImportProjectByFileSpec</a>
<a href="#">CreateProjectByFileSpec</a>	<a href="#">IsBuildInProgress</a>
<a href="#">CreateUserTree</a>	<a href="#">OpenDocument</a>
<a href="#">DoCommand</a>	<a href="#">OpenProject</a>
<a href="#">FindDesignForDataModel</a>	<a href="#">OpenProjectWithOptions</a>
<a href="#">FindLogicalFolder</a>	<a href="#">OpenProjectByFileSpec</a>
<a href="#">get_ActiveDocument</a>	<a href="#">OpenProjectByFileSpecWithOptions</a>
<a href="#">get_Application</a>	<a href="#">OpenTextDocument</a>
<a href="#">get_CompareInterface</a>	<a href="#">OpenTextDocumentByFileSpec</a>
<a href="#">get_CreatableItems</a>	<a href="#">OpenUntitledTextDocument</a>
<a href="#">get_Debugger</a>	<a href="#">put_AllowUserInteraction</a>
<a href="#">get_DefaultProject</a>	<a href="#">put_Visible</a>
<a href="#">get_DefaultProjectDocument</a>	<a href="#">QueueDeferredAction</a>
<a href="#">get_Documents</a>	<a href="#">Quit</a>
<a href="#">get_FullName</a>	<a href="#">RemoveCreatableItem</a>
<a href="#">get_Name</a>	<a href="#">RemoveNamedPluginData</a>
<a href="#">GetNamedPluginData</a>	<a href="#">RemoveUserTree</a>

## Application

*AddCreatableItem*

---

<a href="#">get_Projects</a>	<a href="#">SetNamedPluginData</a>
<a href="#">GetSetting</a>	<a href="#">SetSetting</a>
<a href="#">get_UserTrees</a>	

---

## AddCreatableItem

Use this method to add a creatable item to the CodeWarrior IDE. Creatable items encapsulate the items visible in the **New** window. See [“Creatable Items” on page 113](#) for more information on creatable items.

```
virtual HRESULT AddCreatableItem(
    IUnknown *item) = 0;
```

item

The creatable item to add to the CodeWarrior IDE.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## AddUserTree

This method adds an existing user tree to the application.

```
virtual HRESULT AddUserTree(
    ICodeWarriorUserTree *pval) = 0;
```

pval

A pointer to the user tree to add to the application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorUserTree” on page 26](#)

## AttemptModify

Use this method to request that a CodeWarrior file be made writable.

Prototype `virtual HRESULT AttemptModify(  
    IFileSpec *fileSpec,  
    ECodeWarriorVCSInteractionOption uiParameter,  
    ICodeWarriorProject *project) = 0;`

Parameters `fileSpec`

A pointer to the [IFileSpec](#) interface containing the file in question.

`uiParameter`

A [ECodeWarriorVCSInteractionOption](#) set to a value in the range defined by the [ECodeWarriorVCSInteractionOption](#) enumeration, representing how the IDE should handle user interaction.

`project`

If the file is in a project that has version control enabled, use a pointer to the [ICodeWarriorProject](#) interface. Otherwise, use NULL.

Returns `S_OK` if the file was found and made writable, `S_FALSE` if the file cannot be modified, or `E_ABORT` if the user cancelled the operation.

---

## CreateProject

Use this method to create a new project in the CodeWarrior application by specifying a file path.

```
virtual HRESULT CreateProject(  
    BSTR filePath,  
    BSTR linkerName,  
    BSTR designName,  
    BSTR targetName,
```

```
VARIANT_BOOL fMakeVisible,  
ICodeWarriorProject **pval) = 0;
```

filepath

The full path to the new project file. CodeWarrior creates this file. It should not exist prior to this call.

linkerName

The name of the linker used in this project. CodeWarrior configures the new project to use the linker you specify here. You can set this value to NULL or an empty string to use the default values from the project.

designName

The name of the initial design in this project. CodeWarrior creates a new design with the name specified. You can set this value to NULL or an empty string to use the default values from the project.

targetName

The name of the initial target in this project. CodeWarrior creates a new target with the name specified. You can set this value to NULL or an empty string to use the default values from the project.

fMakeVisible

Set this parameter to `true` if this project should be visible to users or `false` if not.

pval

On return, this parameter contains the address of a pointer to the new project.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

## CreateProjectByFileSpec

Use this method to create a new project in the CodeWarrior application, by file specification.

```
virtual HRESULT CreateProjectByFileSpec(  
    IFileSpec *projectFileSpec,  
    BSTR linkerName,  
    BSTR designName,  
    BSTR targetName,  
    IFileSpec *stationeryFileSpec,  
    VARIANT_BOOL fMakeVisible,  
    ICodeWarriorProject **pval) = 0;
```

projectFileSpec

A pointer to the [IFileSpec](#) interface. CodeWarrior creates this file. It should not exist prior to this call.

linkerName

The name of the linker used in this project. CodeWarrior configures the new project to use the linker you specify here. You can set this value to NULL or an empty string to use the default values from the project.

designName

The name of the initial design in this project. CodeWarrior creates a new design with the name specified. You can set this value to NULL or an empty string to use the default values from the project.

targetName

The name of the initial target in this project. CodeWarrior creates a new target with the name specified. You can set this value to NULL or an empty string to use the default values from the project.

stationeryFileSpec

If this project is to be based on existing stationery, use a pointer to the [IFileSpec](#) interface set to a stationery project file. If not,

use NULL.

fMakeVisible

Set this parameter to `true` if this project should be visible to users or `false` if not.

pval

On return, this parameter contains the address of a pointer to the new project.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)  
["ICodeWarriorProject" on page 216](#)

---

## CreateUserTree

This method creates a new user tree.

```
virtual HRESULT CreateUserTree(  
    BSTR displayName,  
    BSTR value,  
    EUserDefinedTree type,  
    BSTR keyName,  
    ICodeWarriorUserTree **pVal) = 0;
```

displayName

The name of the user tree that will appear in the IDE.

value

The value string of the user tree.

type

The type of the tree, which must be one of the values specified by the [EUserDefinedTree](#) Tree enumeration.



keyName

The key name of the user tree.

pval

On return, this parameter contains the address of a pointer to the new user tree.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EUserDefinedTree” on page 30](#)  
[“ICodeWarriorUserTree” on page 26](#)

---

## DoCommand

Use this method to invoke a command in the CodeWarrior IDE.

```
virtual HRESULT DoCommand(  
    long commandID) = 0;
```

commandID

Set this long value to the command ID of the command to invoke. The command ID must be previously registered with the IDE.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Commands API Overview” on page 83](#)

---

## FindDesignForDataModel

Use this method

```
virtual HRESULT FindDesignForDataModel(  
    IUnknown *dataModel,  
    ICodeWarriorDesign **project) = 0;
```

## Application

### FindLogicalFolder

---

dataModel

Supply a pointer to the IUnknown interface containing the data model corresponding to the design you are looking for.

project

On return, this parameter contains the address of a pointer to the design corresponding to the specified data model.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorDesign" on page 130](#)

---

## FindLogicalFolder

Use this method to obtain a file specification for one of the standard folders used by the CodeWarrior IDE. A list of these folder names is provided under ["Standard Folder Names" on page 74](#).

```
virtual HRESULT FindLogicalFolder(
    BSTR folderName,
    IFileSpec **folder) = 0;
```

folderName

The name of the folder you want. For a list of accepted folder names, see ["Standard Folder Names" on page 74](#).

folder

On return, this parameter contains the address of a pointer to a file specification for the folder in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)

---

## get\_ActiveDocument

Call this method to obtain the currently active document in the CodeWarrior application.

```
virtual HRESULT get_ActiveDocument(  
    ICodeWarriorDocument **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the active document in the CodeWarrior application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorDocument" on page 160](#)

---

## get\_Application

Call this method to obtain the CodeWarrior application object.

```
virtual HRESULT get_Application(  
    IDispatch **pval) = 0;
```

pval

Upon return this parameter contains the address of a pointer to the CodeWarrior application object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Application***get\_CompareInterface*

---

---

**get\_CompareInterface**

This method gives access the comparison dialog, so that the [ICodeWarriorCompare](#) interface can be used to compare files and folders.

```
virtual HRESULT get_CompareInterface(  
    ICodeWarriorCompare **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the comparison information.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorCompare” on page 69](#)

---

**get\_CreatableItems**

Call this method to obtain a collection of all creatable items in the CodeWarrior application.

```
virtual HRESULT get_CreatableItems(  
    ICodeWarriorCreatableItemCollection **pval  
    ) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all creatable items in the CodeWarrior application.

---

**get\_Debugger**

This method gets an object that defines the current debugger.

```
virtual HRESULT get_Debugger(  

```

---

```
ICodeWarriorDebugger **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the object that defines the current debugger.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_DefaultProject

Use this method to obtain the default project object in the CodeWarrior application.

```
virtual HRESULT get_DefaultProject(  
    ICodeWarriorProject **project) = 0;
```

project

On return, this parameter contains the address of a pointer to the default project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

---

## get\_DefaultProjectDocument

Call this method to obtain the default project document in the CodeWarrior application.

```
virtual HRESULT get_DefaultProjectDocument(  
    ICodeWarriorProjectDocument **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the default project document in the CodeWarrior application.

## Application

### *get\_Documents*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProjectDocument" on page 168](#)

---

## get\_Documents

Call this method to obtain a collection of all open documents in the CodeWarrior application.

```
virtual HRESULT get_Documents(
    ICodeWarriorDocumentCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all open documents in the CodeWarrior application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_FullName

Call this method to obtain the full path of the CodeWarrior application file.

```
virtual HRESULT get_FullName(
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the full path to the application file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Name

Call this method to obtain the name of the CodeWarrior application file.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of the application file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetNamedPluginData

Use this method to obtain plug-in data from a given plug-in.

```
virtual HRESULT GetNamedPluginData(  
    BSTR resourceName,  
    IStream **pluginData) = 0;
```

resourceName

The name of the plug-in resource you want to obtain data from.

pluginData

On return, this parameter contains the address of a pointer to the data for plug-in resource specified.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## get\_Projects

Call this method to obtain a collection of all currently open projects in the CodeWarrior application.

```
virtual HRESULT get_Projects(
    ICodeWarriorProjectCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all currently open projects in the CodeWarrior application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetSetting

Use this method to obtain the value of a given IDE preference setting.

```
virtual HRESULT GetSetting(
    BSTR settingsName,
    VARIANT *pval) = 0;
```

settingsName

The name of the setting value to get.

pval

On return, this parameter contains the value of the specified setting.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



## get\_UserTrees

This method gets the user-defined trees, as a collection.

```
virtual HRESULT get_UserTrees(  
    ICodeWarriorUserTreeCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection that holds the user-defined trees.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_VersionControl

Use this method to obtain the version control interface to the CodeWarrior application.

```
virtual HRESULT get_VersionControl(  
    ICodeWarriorVersionControl **vcs) = 0;
```

vcs

On return, this parameter contains the address of a pointer to the version control interface to the CodeWarrior application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorVersionControl” on page 370](#)

---

## get\_Visible

Call this method to obtain the visible state of the CodeWarrior application.

```
virtual HRESULT get_Visible(
    VARIANT_BOOL *pval) = 0;
```

`pval`

On return, this parameter is set to `true` if the application is visible or `false` if the application is not visible.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## ImportProject

Use this method to import an XML project file into the CodeWarrior IDE, specifying the full path to the import file.

```
virtual HRESULT ImportProject(
    BSTR textFilePath,
    BSTR projectFilePath,
    VARIANT_BOOL fMakeVisible,
    ICodeWarriorProject **pval) = 0;
```

`textFilePath`

The full path to the XML file you are importing.

`projectFilePath`

The full path to the new project file. This file must not exist. It is created by CodeWarrior.

`fMakeVisible`

Set this parameter to `true` if this operation is to be visible to the user or `false` if the operation should be hidden from the user.

pval

On return, this parameter contains the address of a pointer to the resulting project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

---

## ImportProjectByFileSpec

Use this method to import an XML project file into the CodeWarrior IDE, specifying a file specification to the import file.

```
virtual HRESULT ImportProjectByFileSpec(  
    IFileSpec *textFileSpec,  
    IFileSpec *projectFileSpec,  
    VARIANT_BOOL fMakeVisible,  
    ICodeWarriorProject **pval) = 0;
```

textFileSpec

A pointer to the [IFileSpec](#) interface containing the file specification for the XML file you are importing.

projectFileSpec

A pointer to the [IFileSpec](#) interface containing the file specification for the new project file. This file must not exist. It is created by CodeWarrior.

fMakeVisible

Set this parameter to `true` if this operation is to be visible to the user or `false` if the operation should be hidden from the user.

pval

On return, this parameter contains the the address of a pointer to the new project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**Application***IsBuildInProgress*

---

See Also ["IFileSpec" on page 185](#)["ICodeWarriorProject" on page 216](#)

---

**IsBuildInProgress**

Use this method to determine if a build is currently in progress in the CodeWarrior IDE.

```
virtual HRESULT IsBuildInProgress(  
    VARIANT_BOOL *pval) = 0;
```

*pval*

On return, this parameter is set to `true` if a build is in progress or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

**OpenDocument**

This method opens a document specified by a full file path.

```
virtual HRESULT OpenDocument(  
    BSTR filePath) = 0;
```

*filePath*

The full path to the document to open.

---

## OpenProject

Use this method to open a project in the CodeWarrior IDE, by specifying the project file by full path.

```
virtual HRESULT OpenProject(  
    BSTR filePath,  
    VARIANT_BOOL fMakeVisible,  
    ECodeWarriorConvertOption convertOption,  
    ECodeWarriorRevertPanelOption revertOption,  
    ICodeWarriorProject **pval) = 0;
```

filePath

The full path to the project file.

fMakeVisible

Set this parameter to `true` if this operation is to be visible to the user or `false` if the operation should be hidden from the user.

convertOption

A value in the range defined by the [ECodeWarriorConvertOption](#) enumeration, representing how the CodeWarrior IDE should handle this project if the project is found to be a project created by an older version of the IDE.

revertOption

A value in the range defined by the enumeration [ECodeWarriorRevertPanelOption](#), representing whether revert is allowed in settings panels of the project being opened.

pval

On return, this parameter contains the address of a pointer to the new project.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorConvertOption” on page 72](#)

[“ECodeWarriorRevertPanelOption” on page 72](#)

[“ICodeWarriorProject” on page 216](#)

---

## OpenProjectWithOptions

Use this method to open a project in the CodeWarrior IDE, by specifying the project file by full path and applying certain options.

```
virtual HRESULT OpenProjectWithOptions(  
    BSTR filePath,  
    VARIANT_BOOL fMakeVisible,  
    ECodeWarriorConvertOption convertOption,  
    ECodeWarriorRevertPanelOption revertOption,  
    ECodeWarriorProjectOption projectOption,  
    ICodeWarriorProject **pval) = 0;
```

`filePath`

The full path to the project file.

`fMakeVisible`

Set this parameter to `true` if this operation is to be visible to the user or `false` if the operation should be hidden from the user.

`convertOption`

A value in the range defined by the [ECodeWarriorConvertOption](#) enumeration, indicating how the CodeWarrior IDE should handle this project if the project is found to be a project created by an older version of the IDE.

`revertOption`

A value in the range defined by the enumeration [ECodeWarriorRevertPanelOption](#), indicating whether revert is allowed in settings panels of the project being opened.

`projectOption`

A value in the range defined by the enumeration [ECodeWarriorProjectOption](#), indicating whether to cache

subprojects when the project is opened.

pval

On return, this parameter contains the address of a pointer to the specified project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ECodeWarriorConvertOption" on page 72](#)

["ECodeWarriorRevertPanelOption" on page 72](#)

["ECodeWarriorProjectOption" on page 73](#)

["ICodeWarriorProject" on page 216](#)

---

## OpenProjectByFileSpec

This method opens a project in the CodeWarrior IDE, by specifying the project file with a file specification.

```
virtual HRESULT OpenProjectByFileSpec(  
    IFileSpec *fileSpec,  
    VARIANT_BOOL fMakeVisible,  
    ECodeWarriorConvertOption convertOption,  
    ECodeWarriorRevertPanelOption revertOption,  
    CodeWarriorProject **pval) = 0;
```

fileSpec

A pointer to the [IFileSpec](#) interface containing the file specification of the project file to open.

fMakeVisible

Set this parameter to `true` if this operation is to be visible or `false` if it is to be hidden from the user.

convertOption

A value in the range defined by the [ECodeWarriorConvertOption](#) enumeration, representing



## **Application**

*OpenProjectByFileSpec*

---

how the CodeWarrior IDE should handle this project if the project is found to be a project created by an older version of the IDE.



revertOption

A value in the range defined by the enumeration [ECodeWarriorRevertPanelOption](#), representing whether revert is allowed in settings panels of the project being opened.

pval

On return, this parameter contains the address of a pointer to the new project.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)

["ECodeWarriorConvertOption" on page 72](#)

["ECodeWarriorRevertPanelOption" on page 72](#)

["ICodeWarriorProject" on page 216](#)

---

## OpenProjectByFileSpecWithOptions

This method opens a project in the CodeWarrior IDE, by specifying the project file with a file specification and applying certain options.

```
virtual HRESULT OpenProjectByFileSpecWithOptions(  
    IFileSpec *fileSpec,  
    VARIANT_BOOL fMakeVisible,  
    ECodeWarriorConvertOption convertOption,  
    ECodeWarriorRevertPanelOption revertOption,  
    ECodeWarriorProjectOption projectOption,  
    ICodeWarriorProject **pval) = 0;
```

fileSpec

A pointer to the [IFileSpec](#) interface containing the file specification of the project file to open.

fMakeVisible

Set this parameter to `true` if this operation is to be visible or

## Application

*OpenProjectByFileSpecWithOptions*

---

false if it is to be hidden from the user.

convertOption

A value in the range defined by the [ECodeWarriorConvertOption](#) enumeration, representing how the CodeWarrior IDE should handle this project if the project is found to be a project created by an older version of the IDE.

revertOption

A value in the range defined by the enumeration [ECodeWarriorRevertPanelOption](#), representing whether revert is allowed in settings panels of the project being opened.

projectOption

A value in the range defined by the enumeration [ECodeWarriorProjectOption](#), indicating whether to cache subprojects when the project is opened.

pval

On return, this parameter contains the address of a pointer to the new project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)

["ECodeWarriorConvertOption" on page 72](#)

["ECodeWarriorRevertPanelOption" on page 72](#)

["ECodeWarriorProjectOption" on page 73](#)

["ICodeWarriorProject" on page 216](#)

## OpenTextDocument

Use this method to open and optionally create a text document in the CodeWarrior IDE by specifying the full path.

```
virtual HRESULT OpenTextDocument(  
    BSTR inPath,  
    VARIANT_BOOL create,  
    ICodeWarriorTextDocument **document) = 0;
```

inPath

The full path for the file to open.

create

Set this parameter to `true` if the CodeWarrior IDE should create the file if it does not exist or `false` otherwise.

document

On return, this parameter contains the address of a pointer to the specified document.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTextDocument" on page 171](#)

---

## OpenTextDocumentByFileSpec

Use this method to open and optionally create a text document in the CodeWarrior IDE specifying the file specification.

```
virtual HRESULT OpenTextDocumentByFileSpec(  
    IFileSpec *fileSpec,  
    VARIANT_BOOL create,  
    ICodeWarriorTextDocument **document) = 0;
```

fileSpec

A pointer to the [IFileSpec](#) interface containing the file to

## Application

*OpenUntitledTextDocument*

---

open.

create

Set this parameter to `true` if the CodeWarrior IDE should create the file if it does not exist or `false` if CodeWarrior should not create the file.

document

On return, this parameter contains the address of a pointer to the document specified.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)  
["ICodeWarriorTextDocument" on page 171](#)

---

## OpenUntitledTextDocument

Use this method to open a new text document window with no associated file in the CodeWarrior IDE.

```
virtual HRESULT OpenUntitledTextDocument(
    ICodeWarriorTextDocument **document) = 0;
```

document

On return, this parameter contains the address of a pointer to the new text document.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTextDocument" on page 171](#)

---

---

## put\_AllowUserInteraction

Use this method to set whether or not the CodeWarrior IDE should allow user interaction.

```
virtual HRESULT put_AllowUserInteraction(  
    VARIANT_BOOL pval) = 0;
```

pval

Set this parameter to `true` if the IDE should allow user interaction or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Visible

Call this method to set the visible state of the CodeWarrior application.

```
virtual HRESULT put_Visible(  
    VARIANT_BOOL val) = 0;
```

val

Set this parameter to `true` if the application is visible or `false` if the application is not visible.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## QueueDeferredAction

Use this method to queue a deferred action in the CodeWarrior IDE.

```
virtual HRESULT QueueDeferredAction(  
    IUnknown *action) = 0;
```

---

## Application

### Quit

---

action

A pointer to the `IUnknown` interface containing the deferred action. The deferred action must be previously registered with the IDE.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Commands API Overview” on page 83](#)

---

## Quit

This method closes the CodeWarrior IDE, applying a save option in the process.

```
virtual HRESULT Quit(
    ECodeWarriorSaveOption val) = 0;
```

val

A value in the range defined by the enumeration [ECodeWarriorSaveOption](#), indicating whether to save all the files open in the IDE, ask the user whether to save, or save none of the files.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorSaveOption” on page 73](#)

---

## RemoveCreatableItem

Use this method to remove a creatable item to the CodeWarrior IDE. Creatable items encapsulate the items visible in the **New** window.

```
virtual HRESULT RemoveCreatableItem(
    IUnknown *item) = 0;
```

item

A pointer to the IUnknown interface, containing the creatable item to be removed from the CodeWarrior IDE.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Creatable Items” on page 113](#)

---

## RemoveNamedPluginData

Use this method to remove the plug-in data for a given plug-in.

```
virtual HRESULT RemoveNamedPluginData(  
    BSTR resourceName)
```

resourceName

The name of the plug-in resource you want to modify.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## RemoveUserTree

This method removes a specified user tree.

```
virtual HRESULT RemoveUserTree(  
    ICodeWarriorUserTree *pval) = 0;
```

pval

A pointer to the user tree to remove.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorUserTree” on page 26](#)

---

## Application

*SetNamedPluginData*

---

### SetNamedPluginData

Use this method to set the plug-in data for a given plug-in.

```
virtual HRESULT SetNamedPluginData(
    BSTR resourceName,
    IStream *pluginData) = 0;
```

*resourceName*

The name of the plug-in resource you want to modify.

*pluginData*

A pointer to the *IStream* interface containing the data to load into the plug-in.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

### SetSetting

Use this method to set the value of a given IDE preference setting.

```
virtual HRESULT SetSetting(
    BSTR settingsName,
    VARIANT pval) = 0;
```

*settingsName*

The name of the setting value to modify.

*pval*

The new value of the specified setting.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.



## ICodeWarriorAppEvents

This interface lets plug-ins respond to certain events in the CodeWarrior IDE. The methods outlined below are called by the IDE when a corresponding event occurs.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">DataModelCreated</a>	<a href="#">ProjectVisible</a>
<a href="#">DataModelLoaded</a>	<a href="#">QueryQuit</a>
<a href="#">Application Data Types</a>	<a href="#">Quit</a>
<a href="#">ProjectOpened</a>	<a href="#">Startup</a>

---

### DataModelCreated

The CodeWarrior IDE calls this method when it creates a data model.

```
virtual HRESULT DataModelCreated(
    IUnknown *dataModel,
    VARIANT_BOOL fFromStorage) = 0;
```

**dataModel**

When the IDE calls this method, this parameter contains the created data model.

**fFromStorage**

When the IDE calls this method, this parameter contains `true` if the data model is from storage or `false` if the data model is not from storage.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Application***DataModelLoaded*

---

**DataModelLoaded**

The CodeWarrior IDE calls this method when it loads a data model.

```
virtual HRESULT DataModelLoaded(  
    IUnknown *dataModel) = 0;
```

`dataModel`

When the IDE calls this method, this parameter contains the loaded data model.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**ProjectOpened**

The CodeWarrior IDE calls this method when it opens a project.

```
virtual HRESULT ProjectOpened(  
    ICodeWarriorProject *project,  
    VARIANT_BOOL fVisible) = 0;
```

`project`

When the IDE calls this method, this parameter contains the opened project.

`fVisible`

When the IDE calls this method, this parameter contains `true` if the project is visible, or `false` if the project is not visible.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

## ProjectVisible

The CodeWarrior IDE calls this method when it makes an invisible project visible.

```
virtual HRESULT ProjectVisible(  
    ICodeWarriorProject *project) = 0;
```

project

When the IDE calls this method, this parameter contains the visible project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

---

## QueryQuit

The CodeWarrior IDE calls this method when it wants to quit. Your plug-in should determine whether it is safe to quit, and return a success or failure code accordingly.

```
virtual HRESULT QueryQuit(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Quit

The CodeWarrior IDE calls this method when it quits.

```
virtual HRESULT Quit(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



## Application

### Startup

---

## Startup

The CodeWarrior IDE calls this method when it starts.

```
virtual HRESULT Startup(void) = 0;
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorCompare

This interface provides methods for comparing files and folders.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CompareFiles</a>	<a href="#">CompareFolders</a>
<a href="#">CompareFilesByFileSpec</a>	

---

## CompareFiles

This method compares the contents of two files, optionally comparing case and white space. The results of the comparison appear in the File Compare Results window in the IDE.

```
virtual HRESULT CompareFiles(  
    BSTR srcFile,  
    BSTR destFile,  
    VARIANT_BOOL ignoreCase,  
    VARIANT_BOOL ignoreSpace) = 0;
```

srcFile

The first file to compare.

destFile

The second file to compare.

ignoreCase

true if you want to ignore case in this comparison or false if you want to compare case.

ignoreSpace

true if you want to ignore white space or false if you want to

## Application

*CompareFilesByFileSpec*

---

compare white space.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## CompareFilesByFileSpec

This method compares the contents of two files, as indicated by file specifications, optionally comparing case and white space. The results of the comparison appear in the File Compare Results window in the IDE.

```
virtual HRESULT CompareFilesByFileSpec(
    IFileSpec *srcFile,
    IFileSpec *destFile,
    VARIANT_BOOL ignoreCase,
    VARIANT_BOOL ignoreSpace) = 0;
```

srcFile

A pointer to the first file to compare.

destFile

A pointer to the second file to compare

ignoreCase

true if you want to ignore case in this comparison or false if you want to compare case.

ignoreSpace

true if you want to ignore white space or false if you want to compare white space.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## CompareFolders

This method compares the contents of two folders, optionally comparing case, white space, files that exist in only one folder or the other, and the contents of text files. The results of the comparison appear in the Folder Compare Results window in the IDE.

```
virtual HRESULT CompareFolders(  
    BSTR srcFolder,  
    BSTR destFolder,  
    VARIANT_BOOL inIgnoreCase,  
    VARIANT_BOOL inIgnoreSpace,  
    VARIANT_BOOL showDifferentFiles,  
    VARIANT_BOOL compareTextFileContents) = 0;
```

srcFile

The first folder to compare.

destFile

The second folder to compare

ignoreCase

true if you want to ignore case in this comparison or false if you want to compare case.

ignoreSpace

true if you want to ignore white space or false if you want to compare white space.

showDifferentFiles

true to show files that exist in one folder but not the other or false to ignore such files.

compareTextFileContents

true to compare the contents of text files within the two folders or false to ignore the contents of text files.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Application Data Types

The following data types are used with the Application API:

- [ECodeWarriorConvertOption](#)
- [ECodeWarriorRevertPanelOption](#)
- [ECodeWarriorSaveOption](#)
- [Standard Folder Names](#)
- [ECodeWarriorVCSInteractionOption](#)

### ECodeWarriorConvertOption

This enumeration is used to describe how to treat a project being opened via the [OpenProject](#) and [OpenProjectByFileSpec](#) methods of [ICodeWarriorApp](#).

**Table 3.1** ECodeWarriorConvertOption enumeration

Constant	Description
kCWConvertYes	Convert the project without user interaction.
kCWConvertNo	Do not convert the project.
kCWConvertAsk	Ask the user whether to convert the project or not.

### ECodeWarriorRevertPanelOption

This enumeration is used to describe whether revert is allowed in the settings panels of a project being opened via the [OpenProject](#) and [OpenProjectByFileSpec](#) methods of [ICodeWarriorApp](#).



**Table 3.2 ECodeWarriorRevertPanelOption enumeration**

Constant	Description
kCWDonotRevertPanel	Do not allow the user to revert settings.
kCWAllowPanelRevert	Allow the user to revert settings.

### ECodeWarriorProjectOption

This enumeration is used to describe whether revert is allowed in the settings panels of a project being opened via the [OpenProjectWithOptions](#) and [OpenProjectByFileSpecWithOptions](#) methods of [ICodeWarriorApp](#).

**Table 3.3 ECodeWarriorProjectOption enumeration**

Constant	Description
kCWNone	Apply the default settings when opening a project.
kCWDisableSubProjectCaching	Disable the caching of sub projects when opening a project.

### ECodeWarriorSaveOption

This enumeration describes settings for saving files when making the CodeWarrior IDE Quit. It is used by the [Quit](#) method of [ICodeWarriorApp](#).

**Table 3.4 ECodeWarriorSaveOption enumeration**

Constant	Description
kCWAskSave	Asks the user whether to save all the files before closing the IDE.
kCWSaveAll	Saves all the files before closing the IDE.
kCWSaveNone	Closes the IDE without saving any files.

### Standard Folder Names

These folder names represent standard folders within the CodeWarrior folder and are for use with the [FindLogicalFolder](#) method of [ICodeWarriorApp](#).

**Table 3.5 Standard folder name enumeration**

Constant	Description
kMWStationeryFolder	The folder where CodeWarrior project stationery is stored
kMWRadStationeryFolder	The folder where CodeWarrior RAD stationery is stored
kMWCompilerFolder	The folder where the CodeWarrior IDE resides
kMWPluginsFolder	The folder where CodeWarrior plug-ins reside
kLocalizedResourcesFolder	The folder where localized resources reside

### ECodeWarriorVCSInteractionOption

This enumeration is used to describe how user interaction should be handled when a version control operation is performed by the CodeWarrior IDE. It is used in the [AttemptModify](#) method of [ICodeWarriorApp](#).

**Table 3.6** ECodeWarriorVCSInteraction enumeration

Constant	Description
kCWAsk	Ask the user whether to make version control changes
kCWDoNothing	Do not make version control changes
kCWUseDefault	Use the default version control behavior when making changes - useful when working with invisible projects



**Application**  
*Application Data Types*

---

# Collections

---

This chapter describes how to use the Collections API to create and manage Collections in the CodeWarrior IDE.

This chapter contains the following sections:

- [Collections API Overview](#)
- [Using the Collections API](#)
- [Collections API Reference](#)

## Collections API Overview

The Collections API is a set of interfaces that allows a plug-in to create and manipulate collections of IDE-related objects. A collection is a class that holds a list of similar items. The CodeWarrior IDE uses collections to hold lists of IDE-related objects.

## Using the Collections API

Most of the collections returned by the IDE are read-only. Calling **Add** or **Remove** methods on them returns `E_FAIL`. The **Add** and **Remove** methods are provided for collections that users create to pass into the IDE.

## Collections API Reference

The Collections API contains numerous interfaces for working with numerous types of data. However, all collections in this API implement the same methods and behaviors. This section describes a single generic collection that applies to all CodeWarrior collections. CodeWarrior collections are provided for each of the data types in [Table 4.1](#).

**Table 4.1 Data Types with Associated Collections**

BSTR
ICodeWarriorAccessPath
ICodeWarriorBaseClass
ICodeWarriorClass
ICodeWarriorComponent
ICodeWarriorComponentEvent
ICodeWarriorComponentEventSet
ICodeWarriorComponentProperty
ICodeWarriorCreatableItem
ICodeWarriorDataMember
ICodeWarriorDesign
ICodeWarriorDocument
ICodeWarriorMessage
ICodeWarriorMethod
ICodeWarriorProject
ICodeWarriorProjectFile
ICodeWarriorSubTarget
ICodeWarriorSymbol
ICodeWarriorTarget
ICodeWarriorTargetFile
ICodeWarriorUserTree
IFileSpec

## CodeWarrior Collections

This is a generic description of all collection interfaces provided by the CodeWarrior COM API. This description applies to all CodeWarrior collection interfaces.

As a rule, the name of each CodeWarrior collection interface is constructed by taking the name of the data interface for the collection (for example, ICodeWarriorAccessPath), and appending the word “Collection” to the end of it (as in ICodeWarriorAccessPathCollection). All of the collection data types are listed in [Table 4.1 on page 78](#).

---

**NOTE** The exception to this rule is IBSTRCollection, where “I” is prepended to the data type name.

---

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

Every CodeWarrior collection interface provides these methods:

<a href="#">Add</a>	<a href="#">get_ReadOnly</a>
<a href="#">get_Count</a>	<a href="#">Item</a>
<a href="#">get_NewEnum</a>	<a href="#">Remove</a>

---

### Add

This method appends an item to a collection. The [get\\_Count](#) method reflects whether or not this method can be used.

```
virtual HRESULT Add(
    VarType *var) = 0;
```

var

One of the collection data types listed in [Table 4.1 on page 78](#).

## Collections

### *get\_Count*

---

**NOTE** For the BSTR collection data type, use a BSTR instead of a pointer to a BSTR.

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### get\_Count

This method gets the number of items in a collection.

```
virtual HRESULT get_Count(
    long *pval) = 0;
```

pval

On return, this parameter contains a pointer to the number of items contained in this collection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### get\_\_NewEnum

This method gets an enumerator for a collection.

```
virtual HRESULT get__NewEnum(
    IDispatch **pval) = 0;
```

pval

Upon return, this parameter the address of a pointer to an enumerator for this collection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---



---

## get\_ReadOnly

This method gets the read-only status of a collection. The result of this method determines whether the **Add** and **Remove** methods of a collections may be used.

**NOTE** All CodeWarrior-generated collections are read only. Only user-generated collections can be modified.

---

```
virtual HRESULT get_ReadOnly(  
    VARIANT_BOOL *bool) = 0;
```

bool

On return, this parameter is set to `true` if the collection is read-only or `false` if the collection is modifiable.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Item

This method gets an item from a collection.

```
virtual HRESULT Item(  
    long index,  
    VarType *var) = 0;
```

index

The index of the item you want.

var

One of the collection data types listed in [Table 4.1 on page 78](#). On return, this parameter contains a pointer to the requested item.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Collections***Remove*

---

---

**Remove**

This method removes an item from a collection. The [get\\_Count](#) method reflects whether or not this method can be used.

```
virtual HRESULT Remove(  
    VarType *var) = 0;
```

var

One of the collection data types listed in [Table 4.1 on page 78](#).

---

**NOTE** For the BSTR collection data type, use a BSTR instead of a pointer to a BSTR.

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

# Commands

---

This chapter shows how to use the Command API to allow plug-ins to intercept commands from the CodeWarrior IDE.

This chapter contains the following sections:

- [Commands API Overview](#)
- [Using the Commands API](#)
- [Commands API Reference](#)

## Commands API Overview

The Command API allows plug-ins to respond to commands from the CodeWarrior IDE.

The commands interface allows users to define command groups and assign certain command features at initialization and during run time.

All interfaces defined in the commands API are pure abstract base classes. All interfaces defined in this section inherit from `IUnknown`.

Command groups are created only when the IDE launches. Run time manipulation of commands is achieved via `ICodeWarriorCommandHandler::GetCommandStatus()`.

## Using the Commands API

Creating a command group to intercept user events within the IDE typically involves several steps and can involve several interfaces. However, some of the most common uses of command groups are for creating menus and menu items at IDE launch and enabling menu items to perform desired actions. Typical steps for creating a command group, such as menus, involve:

- [Creating a Command Group](#)
- [Assigning a Command Handler](#)
- [Registering a Command](#)
- [Displaying a Command Group](#)

The Commands API also uses data types, as described in this section:

- [Commands Data Types](#)

### Creating a Command Group

All commands are created with the `ICodeWarriorCommandRegistry` interface. However, you must first provide a service provider interface (`IServiceProvider`) before you can create a command group. You also must know the GUID of the CodeWarrior Service ID as well as the Interface ID. [Listing 5.1](#) demonstrates how to use the `QueryService()` method when creating a new command group.

#### Listing 5.1 Example Code - Creating a Command Group

---

```
ICodeWarriorCommandRegistry *cmdRegistry;

// Here we query the service provider to get the command registry
// interface.

servProv->QueryService(
    SID_SCodeWarriorCommandRegistry,
    IID_ICodeWarriorCommandRegistry, &cmdRegistry);

// If the command is supported by the IDE then we can create
// our command info and register our commands
```

```

if( cmdRegistry ){

    // Assign command handlers if we want to here

    // Create the command group here, which will be used for our
    // menu with the title "My Menu"
    BSTR bstr = SysAllocString( OLESTR("My Menu"));
    cmdRegistry->CreateNewCommandGroup( kToolbarTestPluginID,
        cmdGroup_TestPlugin, bstr, cmdGroup_Nothing );

    // Get toolbar icon info if any

    // Register Commands

    // Release
}

```

---

### Assigning a Command Handler

Assigning a command handler to a command group item allows that item to respond to events handled from a particular command. We can create a reference to an interface by creating a command handler and then registering it in the `commandHandler` field of the `SRegisterCommandInfo` structure. [Listing 5.2](#) shows how to assign a command handler that will be used for a menu item, which is demonstrated in [Listing 5.3](#).

#### Listing 5.2 Example Code - Assigning a Command Handler

---

```

ICodeWarriorWindowManager *windowMgr = NULL;

servProv->QueryService(SID_SCodeWarriorWindowManager,
    IID_ICodeWarriorWindowManager, &windowMgr);

/* Handle commands for creating a new window */
ICodeWarriorCommandHandler *cmdHandler = new
    ExampleCommandHandler(windowMgr);

```

---

### Registering a Command

Once you have a command group, you can register individual commands. All commands should be registered with the

## Commands

### Using the Commands API

---

`RegisterCommand()` method from the `ICodeWarriorCommandHandler` interface. The `commandHandler` field of the `SRegisterCommandInfo` structure contains a pointer to the interface whose routines you want to access for this object. [Listing 5.2](#) shows how to assign a command handler for a command group item.

In order to respond to commands sent by the IDE, your plug-in must register a command handler for the particular command (`commandID`) you want the plug-in group (`commandGroupID`) to intercept within the main plug-in (`pluginID`). An example of using the command registry is shown in [Listing 5.3](#).

#### Listing 5.3 Example Code - Registering a Command

---

```
// A new menu item identified by cmd_NewPluginWindow will be
// added to the command group specified by cmdGroup_TestPlugin
// You will need to make a similar call for each item in you
// menu.

SysReAllocString(&bstr, OLESTR("My New Window"));
SRegisterCommandInfo cmdInfo;
cmdInfo.pluginID = kToolbarTestPluginID;
cmdInfo.commandID = cmd_NewPluginWindow;
cmdInfo.commandGroupID = cmdGroup_TestPlugin;
cmdInfo.commandName = bstr;
cmdInfo.toolbarIcon = tbIconInfo;
cmdInfo.visibleInMenu = true;
cmdInfo.itemType = CWCommandItemType_Command;
cmdInfo.extraInfo.commandHandler = cmdHandler;
cmdRegistry->RegisterCommand(cmdInfo, cmd_Nothing);

// Always call Release when you are through with an interface!
cmdRegistry->Release();
}
```

---

#### Displaying a Command Group

The last step in establishing your command group is displaying it within the IDE so the user can access it. [Listing 5.4](#) demonstrates how to display a command group as a menu on the menu bar at IDE launch.

**Listing 5.4 Example Code - Displaying a Command Group**

---

```
// Show the command group in a menu
// We will do this by providing a menu manager interface to
// manage our commands

ICodeWarriorMenuManager *menuMgr;
servProv->QueryService(SID_SCodeWarriorMenuManager,
    IID_ICodeWarriorMenuManager, &menuMgr );
if ( menuMgr ){
    menuMgr->ShowCommandGroupMenu(kToolbarTestPluginID,
        cmdGroup_TestPlugin, true);
    menuMgr->Release();
}
```

---

## Commands API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorCommandHandler](#)
- [ICodeWarriorCommandRegistry](#)
- [ICodeWarriorDeferredAction](#)

The following data types are used with these interfaces:

- [Command status](#)
- [SRegisterCommandGroup](#)



## ICodeWarriorCommandHandler

This interface allows you to intercept and handle built-in commands as well as custom commands registered by your plug-in.

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

---

[ExecuteCommand](#)[GetCommandStatus](#)

---

---

## ExecuteCommand

The IDE calls this method when a command needs to be executed by your plug-in. Usually this is the result of an associated menu or toolbar button being selected by the user.

```
virtual HRESULT ExecuteCommand(  
    CWCommandID inCommandNumber,  
    ICodeWarriorCommandHandler *inDefaultHandler  
    ) = 0;
```

`inCommandNumber`

The ID of a command handler to be executed. Use a built-in command handler or an external command handler registered by the plug-in via the [RegisterCommand](#) method. To handle a built-in command, test this parameter to see if it is equal to the appropriate predefined constant in `CodeWarriorCommandNumbers.h`.

`inDefaultHandler`

The default command handler for this command. If the plug-in does not handle this command, it should call this handler to pass control on to that handler.

**Commands***GetCommandStatus*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorCommandHandler” on page 89](#)

[“RegisterCommand” on page 93](#)

---

## GetCommandStatus

This method is called to update menu items at run-time.

```
virtual HRESULT GetCommandStatus(  
    CWCommandID inCommandNumber,  
    BOOL &outEnabled, SH  
    ORT &outCheckedState,  
    BSTR &outNewName,  
    ICodeWarriorCommandHandler *inDefaultHandler  
    ) = 0;
```

*inCommandNumber*

The ID of the command handler to be executed. This may be a built-in command handler or a command handler that was registered by the plug-in.

*outEnabled*

The current enabled state of this command. If the command is enabled, this parameter returns true. Otherwise, it returns false. The visual element for this command in the CodeWarrior IDE is drawn accordingly.

*outCheckedState*

The current checked state of the menu item for this command. If the item is checked, this parameter returns true. Otherwise, it returns false.

*outNewName*

The current name of this command, as it is displayed in the IDE menus or the toolbar tool tips. Set this item to NULL for no change of the current menu item title.

---

`inDefaultHandler`

The default command handler for this command. If the plug-in does not handle this command, it should call this handler to pass control on to that handler.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Enumeration for Command Status” on page 95](#)

## ICodeWarriorCommandRegistry

This interface allows you to create and dispose of your own custom commands in the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CreateNewCommandGroup</a>	<a href="#">RegisterCommand</a>
---------------------------------------	---------------------------------

---

---

## CreateNewCommandGroup

This method creates a new command group. A command group is a menu or sub menu. The constant `cmdGroup_Nothing` is used to define a new menu. You will need to call [ICodeWarriorMenuManager::ShowCommandGroupMenu\(\)](#) to display the menu.

```
virtual HRESULT CreateNewCommandGroup(  
    const CWPluginID inPluginID,  
    CWCommandGroupID inGroupID,  
    BSTR inGroupName,  
    CWCommandGroupID inParentGroupID) = 0;
```

`inPluginID`

The ID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inGroupID`

The ID for the group.

`inGroupName`

The name of the group you want to create (that is, the title of the menu).

`inParentGroupID`

The ID of the parent group.

Returns `HRESULT`

See Also ["ICodeWarriorMenuManager" on page 198](#)

---

## RegisterCommand

This method registers an external command with the CodeWarrior IDE.

```
virtual HRESULT RegisterCommand(
    const SRegisterCommandInfo &inCmdInfo,
    LONG inInsertBeforeCommandID) = 0;
```

### [SRegisterCommandGroup](#)

This data structure must be filled in with the appropriate IDs, names, and types to be registered as a command.

`inInsertBeforeCommandID`

Specifies the insertion location where your command appears. This is typically `cmd_Nothing`, which is defined in `CodeWarriorCommandNumbers.h`.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also ["SRegisterCommandGroup" on page 96](#)

---

## ICodeWarriorDeferredAction

`CodeWarriorDeferredAction.h` defines this interface. The method in this interface can be posted on an application-level queue for execution after the handling of the current event.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">Execute</a>	
-------------------------	--

---



---

## Execute

This method executes a command.

```
virtual HRESULT Execute(void) = 0;
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Commands Data Types

### Command status

The following enumerations in [Table 5.1](#) are used as constants to return the `outCheckedState` parameter of the [ICodeWarriorCommandHandler::GetCommandStatus\(\)](#) method:

**Table 5.1 Enumeration for Command Status**

Constant	Description
<code>CWCommand_CheckMark_NoChange</code>	The specified menu has not changed.
<code>CWCommand_CheckMark_Clear</code>	The specified menu is to be cleared.
<code>CWCommand_CheckMark_Set</code>	The specified menu is set.

### Menu Commands

The following enumeration in [Table 5.2](#) are used to fill out the `itemType` field of the [SRegisterCommandGroup](#) structure when using the [RegisterCommand\(\)](#) method:

**Table 5.2 Enumeration for menu commands**

Constant	Description
<code>CWCommandItemType_Command</code>	The item is a command
<code>CWCommandItemType_Separator</code>	The item is a separator on a menu
<code>CWCommandItemType_SubMenu</code>	The item is a sub menu

## SRegisterCommandGroup

The `SRegisterCommandGroup` structure is used to register a command and set its properties as a parameter of `ICodeWarriorCommandRegistry::RegisterCommand()`. See [Listing 5.3](#) for an example of registering a command with `SRegisterCommandGroup`. This structure is defined as follows:

```
struct SRegisterCommandInfo {
    CWPluginID          pluginID;
    CWCommandID        commandID;
    CWCommandGroupID   commandGroupID;
    BSTR               commandName;
    CWToolbarIconInfo  toolbarIcon;
    BOOL               visibleInMenu;
    long               itemType;
    union {
        IUnknown        *commandHandler;
        CWCommandGroupID subGroupID;
    } extraInfo;
};
```

pluginID

The plug-in ID for the plug-in to which the command belongs.

commandID

The ID of the command group provides the location of the command within a command group specified by `commandGroupID`. The `commandID` values must be in the range of 10,000 to 10,999.

commandGroupID

The command group (menu) to which the item belongs.

commandName

The name of the command to be displayed, as specified by `commandID`.



`toolbarIcon`

A toolbar icon reference for the command you are registering. Pass `toolbarIcon_None` for no icon information. This is a platform specific data type.

On the Mac OS, the `CWToolbarIconInfo` is a `Handle` to the item. Mac users need to provide Mac Toolbox calls to get this resource, which are small icon sweet resoures (`ics#`). See `UseResFile()` and `GetIconSuite()` in the Mac Toolbox for more information.

Windows icons must be registered first before they are used. On Windows, icons are identified by an index into bitmaps that are registered via `ICodeWarriorToolBarRegistry::RegisterToolbarIcons()`.

`visibleInMenu`

Set this item to `true` to display the menu item or `false` to hide the menu item from the user.

`itemType`

Specify the type of item in the command group. Use the enumerations specified in [Table 5.2](#).

`commandHandler`

A pointer to the `IUnknown` interface whose methods you want to use. Set this item to `NULL` if you do not need any additional interface routines. Otherwise, you will need to pass a pointer to the interface whose methods you want to access from the command specified in `commandID`.

`subGroupID`

A sub menu group ID, if one exists.



## **Commands**

*Commands Data Types*

---

# Components

---

This chapter shows how to use the Components API to add your own components in the CodeWarrior IDE.

This chapter contains the following sections:

- [Components API Overview](#)
- [Components API Reference](#)

## Components API Overview

The Components API is a set of interfaces that allows a plug-in to create and manipulate components in the CodeWarrior IDE.

## Components API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorComponent](#)
- [ICodeWarriorComponentEvent](#)
- [ICodeWarriorComponentEventSet](#)
- [ICodeWarriorComponentProperty](#)

## ICodeWarriorComponent

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">get_CanHaveMultipleEventSets</a>	<a href="#">get_EventSets</a>
<a href="#">get_Class</a>	<a href="#">get_Methods</a>
<a href="#">get_DefaultEvent</a>	<a href="#">get_Properties</a>
<a href="#">get_EventConnectionsEnabled</a>	

---

### get\_CanHaveMultipleEventSets

This method gets whether this component can have multiple event sets.

```
virtual HRESULT get_CanHaveMultipleEventSets(
    BOOL *pval) = 0;
```

pval

On return, this parameter is set to `true` if the component can have multiple event sets or `false` if the component cannot have multiple event sets.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## get\_Class

The IDE calls this method to obtain the class of this component.

```
virtual HRESULT get_Class(  
    ICodeWarriorClass **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the class of this component.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorClass" on page 254](#)

---

## get\_DefaultEvent

This method gets the default event for this component.

```
virtual HRESULT get_DefaultEvent(  
    ICodeWarriorComponentEvent **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the default event for this component.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorComponentEvent" on page 104](#)

**Components***get\_EventConnectionsEnabled*

---

---

**get\_EventConnectionsEnabled**

This method gets whether event connections are enabled for this component.

```
virtual HRESULT get_EventConnectionsEnabled(  
    BOOL *pval) = 0;
```

pval

On return, this parameter is set to `true` if event connections are enabled for this component or `false` if event connections are not enabled.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**get\_EventSets**

This method gets the events sets for this component.

```
virtual HRESULT get_EventSets(  
    ICodeWarriorComponentEventSetCollection  
    **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all the methods for this component.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## get\_Methods

This method gets all of the methods for this component.

```
virtual HRESULT get_Methods(  
    ICodeWarriorMethodCollection **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all the methods for this component.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_Properties

This method gets all of the properties for this component.

```
virtual HRESULT get_Properties(  
    ICodeWarriorComponentPropertyCollection  
    **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a collection of all the properties for this component.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## ICodeWarriorComponentEvent

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">GetDefaultMethodName</a>	<a href="#">get Method</a>
<a href="#">get EventSet</a>	<a href="#">get Name</a>

---

### GetDefaultMethodName

This method gets the default method name for this component.

```
virtual HRESULT GetDefaultMethodName(
    IUnknown *modelobject,
    BSTR *pdefname) = 0;
```

modelobject

A pointer to the current object.

pdefname

On return, this parameter contains the default method name for this component.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.



## get\_EventSet

This method gets the component event set that this component event belongs to.

```
virtual HRESULT get_EventSet(  
    ICodeWarriorComponentEventSet **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the event set that this component event belongs to.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorComponentEventSet" on page 107](#)

---

## get\_Method

This method gets the method for this component event.

```
virtual HRESULT get_Method(  
    ICodeWarriorMethod **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the method for this component event.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorMethod" on page 264](#)

## Components

*get\_Name*

---

### get\_Name

This method gets the name of this component event.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

The name of this component event.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorComponentEventSet

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">get_Class</a>	<a href="#">get_EventSetName</a>
<a href="#">get_Events</a>	

---

### get\_Class

This method gets the class for this component event set.

```
virtual HRESULT get_Class(  
    ICodeWarriorClass **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the class for this component event set.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorClass" on page 254](#)

---

### get\_Events

This method gets a collection of the events in this component event set.

```
virtual HRESULT get_Events(  
    ICodeWarriorComponentEventCollection **pval  
    ) = 0;
```

## Components

### *get\_EventSetName*

---

pval

On return, this parameter contains the address of a pointer to the collection of events for this component event set.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

### get\_EventSetName

This method gets the name of this component event set.

```
virtual HRESULT get_EventSetName(
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of this component event set.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorComponentProperty

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">get_Getter</a>	<a href="#">get_Setter</a>
<a href="#">get_Name</a>	<a href="#">get_Type</a>

---

### get\_Getter

This method gets the component method that is responsible for getting this component property.

```
virtual HRESULT get_Getter(  
    ICodeWarriorMethod **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the component method that is responsible for getting this component property.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorMethod" on page 264](#)

## Components

### *get\_Name*

---

### get\_Name

This method gets the name of this component property.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of this component property.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### get\_Setter

This method gets the component method that is responsible for setting this component property.

```
virtual HRESULT get_Setter(  
    ICodeWarriorMethod **pval) = 0;
```

pval

Supply the address of a pointer to the interface. Upon return it contains the component method that is responsible for setting this component property.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorMethod" on page 264](#)

---

## get\_Type

This method gets the type of this component property.

```
virtual HRESULT get_Type(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the type of this component property.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



## Components

*get\_Type*

---



# Creatable Items

---

This chapter shows how to use the Creatable Items API to implement your own creatable items for use in the CodeWarrior IDE.

## Creatable Items API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorCreatableItem](#)
- [ICodeWarriorCreateFileItem](#)
- [ICodeWarriorCreateObjectItem](#)
- [ICodeWarriorCreateProjectItem](#)

These interfaces use the data types described in the following section:

- [Creatable Items Data Types](#)

## ICodeWarriorCreatableItem

This interface defines an item that fits into one of the panes of creatable items visible in the **New** window in the CodeWarrior IDE. Creatable items represent stationery or wizards that the user may use to start a project, file or some other item in the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">GetCategory</a>	<a href="#">GetIcon</a>
<a href="#">GetDisplayName</a>	<a href="#">InvokesWizard</a>

---

## GetCategory

The CodeWarrior IDE calls this method to get the category of this creatable item, which determines where the creatable item is displayed in the **New** window.

```
virtual HRESULT GetCategory(  
    BSTR *category) = 0;
```

category

Set this parameter to one of the creatable item category constants already defined for you, representing the category of this creatable item.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Creatable Item Category Constants” on page 127](#)

## GetDisplayName

The CodeWarrior IDE calls this method to get the display name for this creatable item. The string you supply is used to display the name of this creatable item in the **New** window.

```
virtual HRESULT GetDisplayName(  
    BSTR *displayName) = 0;
```

displayName

Set this string to the name of this creatable item as it should appear in CodeWarrior windows and dialog boxes.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetIcon

The CodeWarrior IDE calls this method to get the icon for this creatable item. The icon you supply is displayed next to this creatable item in the **New** window.

```
virtual HRESULT GetIcon(  
    IUnknown *iconList,  
    int *index) = 0;
```

iconList

A pointer to the IUnknown interface containing the icon list that holds the icon for this creatable item.

index

Set this integer to one of the predefined icon index values or, if your creatable item uses a custom icon, supply the index in the icon list of the custom icon.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Creatable Items***InvokesWizard*

---

See Also [“Built-in Icon Index Values” on page 128](#)

---

**InvokesWizard**

The CodeWarrior IDE calls this method to discover whether this creatable item invokes a wizard or not. If it does invoke a wizard, the IDE appends the localized string for “Wizard” to the display name of the creatable item (for example, "Java Applet Wizard").

```
virtual HRESULT InvokesWizard(void) = 0;
```

Returns S\_OK if creating the item invokes a wizard, or S\_FALSE if creating the item does not invoke a wizard.

## ICodeWarriorCreateFileItem

This interface is used for a creatable file item that is displayed in the **File** pane of creatable items visible in the **New** window in the CodeWarrior IDE.

### Inherited Interfaces

- [ICodeWarriorCreatableItem](#)
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CanAddFileToProject</a>	<a href="#">CreateAndAddFile</a>
<a href="#">CanCreateUntitledFile</a>	<a href="#">CreateUntitledFile</a>

---

### CanAddFileToProject

The CodeWarrior IDE calls this method to determine if this creatable item is able to add a file to a project.

```
virtual HRESULT CanAddFileToProject(void) = 0;
```

Returns S\_OK if this creatable item is able to add files to a project or an appropriate error if this creatable item is not able to add files to a project.

---

### CanCreateUntitledFile

The CodeWarrior IDE calls this method to determine if this creatable item is able to create an untitled file.

```
virtual HRESULT CanCreateUntitledFile(void) = 0;
```

## Creatable Items

### CreateAndAddFile

---

Returns S\_OK if this creatable item is able to create an untitled file or an appropriate error if this creatable item is not able to create an untitled file.

---

## CreateAndAddFile

This method is called by the CodeWarrior IDE when the user instructs it to create a new file with this creatable item selected, and the **Add to project** option is checked.

```
virtual HRESULT CreateAndAddFile(  
    IFileSpec *newFileSpec,  
    ICodeWarriorProject *project,  
    ICodeWarriorTargetCollection *targets,  
    VARIANT_BOOL *fileAdded) = 0;
```

newFileSpec

A pointer to the [IFileSpec](#) interface that contains the file specification for the newly created file.

project

A pointer to the [ICodeWarriorProject](#) interface containing the project to which the new file is to be added.

targets

A pointer to the [ICodeWarriorTargetCollection](#) interface containing a list of the targets to which the new file is to be added.

fileAdded

Set this parameter to `true` if the file is successfully added to the specified project or `false` if the file could not be added.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)

["ICodeWarriorProject" on page 216](#)

[“Using the Collections API” on page 77](#)

---

## CreateUntitledFile

The CodeWarrior IDE calls this method when the user instructs the IDE to create a new file with this creatable item selected, and the **Add to project** option is unchecked.

```
virtual HRESULT CreateUntitledFile(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorCreateObjectItem

This interface is used for a creatable file item that is displayed in the **Object** pane of creatable items visible in the **New** window in the CodeWarrior IDE.

### Inherited Interfaces

- [ICodeWarriorCreatableItem](#)
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">AreObjectsCreatedInDesign</a>	<a href="#">CreateObjectInTargets</a>
<a href="#">CreateObjectInDesign</a>	<a href="#">NeedsObjectName</a>

---

### AreObjectsCreatedInDesign

The CodeWarrior IDE calls this method to determine if this creatable item creates objects in designs.

```
virtual HRESULT AreObjectsCreatedInDesign(
    void) = 0;
```

Returns Return `S_OK` if this creatable item is able to create objects in designs or an appropriate error if this creatable item is not able to create objects in designs.



## CreateObjectInDesign

The CodeWarrior IDE calls this method to instruct this creatable item to create an object in a specific design of a project.

```
virtual HRESULT CreateObjectInDesign(  
    BSTR newItemName,  
    ICodeWarriorProject *project,  
    ICodeWarriorDesign *design) = 0;
```

`newItemName`

The requested name of the new object being created.

`project`

A pointer to the [ICodeWarriorProject](#) interface. This parameter specifies the project containing the design to which the object is being added.

`design`

A pointer to the [ICodeWarriorDesign](#) interface. This parameter specifies the design to which the object is being added.

**Returns** S\_OK if this method call succeeded or an appropriate error if it failed.

**See Also** ["ICodeWarriorProject" on page 216](#)

["ICodeWarriorDesign" on page 130](#)

---

## CreateObjectInTargets

The CodeWarrior IDE calls this method to instruct this creatable item to create an object in a specific target of a project.

```
virtual HRESULT CreateObjectInTargets(  
    BSTR newItemName,  
    ICodeWarriorProject *project,  
    ICodeWarriorTargetCollection *targets) = 0;
```

*newItemName*

The requested name for the new object being created.

*project*

A pointer to the [ICodeWarriorProject](#) interface. This parameter specifies the project containing the design to which the object is being added.

*targets*

A pointer to the [ICodeWarriorTargetCollection](#) interface. This parameter specifies a list of the targets to which the object is being added.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

["Using the Collections API" on page 77](#)

---

## NeedsObjectName

The CodeWarrior IDE calls this method to determine if this creatable item requires a user-specified name. If so, the IDE enables the appropriate option in the **New** dialog..

```
virtual HRESULT NeedsObjectName(void) = 0;
```

Returns **S\_OK** if this creatable item requires a user-specified name or an appropriate error if this creatable item does not require a user-specified name.

## ICodeWarriorCreateProjectItem

This interface is used for a creatable file item that is displayed in the **Project** pane of creatable items visible in the **New** window in the CodeWarrior IDE.

### Inherited Interfaces

- [ICodeWarriorCreatableItem](#)
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CreateInExistingProject</a>	<a href="#">GetCreatedProjectType</a>
<a href="#">CreateNewProject</a>	<a href="#">RequiresFileExtension</a>

---

### CreateInExistingProject

The CodeWarrior IDE calls this method when the user instructs CodeWarrior to create a new project using this creatable item with the **Add to project** option selected in the **New** window. Your plug-in is expected to create a new project and add it to an existing project as a subproject.

```
virtual HRESULT CreateInExistingProject(
    BSTR newItemName,
    ICodeWarriorProject *project) = 0;
```

*newItemName*

A BSTR containing the name of the new project being created.

*project*

A pointer to the [ICodeWarriorProject](#) interface containing the project to which the new project is being added.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

---

## CreateNewProject

The CodeWarrior IDE calls this method when the user instructs IDE to create a new project using this creatable item.

```
virtual HRESULT CreateNewProject(  
    IFileSpec *newFileSpec) = 0;
```

newFileSpec

A pointer to the [IFileSpec](#) interface containing a file specification with the user-specified name of the new project and pointing to the location where the new project should be created.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["IFileSpec" on page 185](#)

---

## GetCreatedProjectType

The CodeWarrior IDE calls this method to determine what type of project this creatable item generates.

```
virtual HRESULT GetCreatedProjectType(  
    ECreateProjectType *pval) = 0;
```

pval

A pointer to a value in the range defined by the [ECreateProjectType](#) enumeration, reflecting the type of project this creatable item generates.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ECreateProjectType" on page 128](#)

---

---

## RequiresFileExtension

The CodeWarrior calls this method to determine if this creatable item requires a file extension.

```
virtual HRESULT RequiresFileExtension(  
    VARIANT_BOOL *pval) = 0;
```

`pval`

Set this parameter to `true` if this creatable item requires a file extension (such as `.mcp` or `.txt`) for the file to be valid. Set it to `false` if this creatable item does not require a file extension. For example, a Mac OS plug-in might return `false`, because a file extension is not required for the file to be valid on Mac OS.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Creatable Items Data Types

The following data types are used with the Creatable Items API:

- [Creatable Item Category Constants](#)
- [Built-in Icon Index Values](#)
- [ECreateProjectType](#)

### Creatable Item Category Constants

These constants are used to describe the possible categories of creatable items displayed by the CodeWarrior IDE in the New window. Implementations of the [ICodeWarriorCreatableItem](#) interface should return one of these constants for their category.




**Table 7.1**    **Creatable Item Categories**

Constant	Description
kMWNNewProjectCategoryName	Creatable items with this category are displayed in the “Project” pane.
kMWNNewFileCategoryName	Creatable items with this category are displayed in the “File” pane.
kMWNNewObjectCategoryName	Creatable items with this category are displayed in the “Object” pane.

### Built-in Icon Index Values

CodeWarrior supplies built-in icons for use with creable items. [Table 7.2](#) shows the indexes of the built-in icons for creable items.

**Table 7.2 Built-in Icons**

Constant	Icon	Value	Description
newIconProject		-1	CodeWarrior project files
newIconTextFile		-2	CodeWarrior text files
newIconCatalog		-3	CodeWarrior catalog files

### ECreateProjectType

CodeWarrior categorizes [ICodeWarriorCreateProjectItem](#) interfaces with the following types based on the capabilities of the creable item in question.

**Table 7.3 ECreateProjectType Enumerations**

Constant	Description
createsProjectOnly	Creable item only creates projects.
createsDesign	Creable item creates and designs.
createsTargets	Creable item creates projects, designs, and targets.



# Designs

---

This chapter shows how to use the Designs API to manage designs in the CodeWarrior IDE.

This chapter contains the following sections:

- [Designs API Overview](#)
- [Designs API Reference](#)

## Designs API Overview

This API lets you manipulate designs within a project. You can use it to add and remove files and targets, initialize and close designs, and otherwise change the details of designs.

## Designs API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorDesign](#)
- [ICodeWarriorDesignAttachment](#)
- [ICodeWarriorDesignEvents](#)

The Design API interfaces use data types defined in the following section:

- [Data Types](#)

## ICodeWarriorDesign

This interface provides methods for working with designs within the CodeWarrior IDE.

### Inherited Interfaces

- IDispatch
- IUnknown

### Methods

This interface provides the following methods:

<a href="#">AddAttachment</a>	<a href="#">get_DataModel</a>
<a href="#">AddFile</a>	<a href="#">get_Name</a>
<a href="#">AddFileByFileSpec</a>	<a href="#">get_Project</a>
<a href="#">CompileFiles</a>	<a href="#">get_Targets</a>
<a href="#">CompileFilesAndWaitToComplete</a>	<a href="#">put_Name</a>
<a href="#">ContainsTarget</a>	<a href="#">RemoveAttachment</a>
<a href="#">FindAndAddFile</a>	<a href="#">RemoveTargetFromDesign</a>
<a href="#">get_BrowserDB</a>	

---

## AddAttachment

This method adds an attachment to the design.

```
virtual HRESULT AddAttachment(
    const CLSID *attachmentCLSID) = 0;
```

attachmentCLSID

A pointer to the attachment.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“RemoveAttachment” on page 139](#)

## AddFile

This method adds a file to the design.

```
virtual HRESULT AddFile(  
    BSTR path,  
    BSTR groupPath,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

The absolute path to the file you want to add to the design.

groupPath

The absolute path to the group within which the new file should be added.

projectFile

The address of a pointer to the file you want to add to the project.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## AddFile2

This method adds a file to the design and set link flags on the file.

```
virtual HRESULT AddFile2(  
    BSTR path,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

The absolute path to the file you want to add to the design.

groupPath

The path to the group within which the new file should be added.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

projectFile

The address of a pointer to the file you want to add to the project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

---

## AddFile2ByFileSpec

This method adds a file to the design, by using a file specification object, and set link flags on the file.

```
virtual HRESULT AddFile2ByFileSpec(
    IFileSpec *fileSpec,
    BSTR groupPath,
    ECodeWarriorLinkFlags linkFlags,
    ICodeWarriorProjectFile **projectFile) = 0;
```

fileSpec

The file specification (as a pointer to an IFileSpec object) that defines the file to add to the project.

groupPath

The absolute path to the group within which the new file should be added.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#)

enumeration, representing how the linker should link this file.

projectFile

The address of a pointer to the file you want to add to the project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## AddFileByFileSpec

This method adds a file to the design, by using a file specification object.

```
virtual HRESULT AddFileByFileSpec(  
    IFileSpec *fileSpec,  
    BSTR groupPath,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

fileSpec

The file specification (as a pointer to an IFileSpec object) that defines the file to add to the project.

groupPath

The absolute path to the group within which the new file should be added.

projectFile

The address of a pointer to the file you want to add to the project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## CompileFiles

This method causes a compilation of the current design.

```
virtual HRESULT CompileFiles(
    ICodeWarriorProjectFileCollection *collection,
    long *cookie) = 0;
```

collection

A pointer of type ICodeWarriorProjectFileCollection indicating the collection of files to compile.

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## CompileFilesAndWaitToComplete

This method causes a compilation of the current design and returns all the build messages created by the compiler.

```
virtual HRESULT CompileFilesAndWaitToComplete(
    ICodeWarriorProjectFileCollection *collection,
    ICodeWarriorBuildMessages **buildMessages) = 0;
```

collection

A pointer of type ICodeWarriorProjectFileCollection indicating the collection of files to compile.

buildMessages

On return, this parameter contains the address of a pointer to the build messages generated by the compiler.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## ContainsTarget

This method discovers if the current design contains a particular target.

```
virtual HRESULT ContainsTarget(  
    ICodeWarriorTarget *target) = 0;
```

target

A pointer to the target you are looking for within the design.

Returns S\_OK if the specified target is present within the current design or an appropriate error if not.

---

## FindAndAddFile

This method finds a file and adds it to the design.

```
virtual HRESULT FindAndAddFile(  
    BSTR path,  
    BSTR groupPath,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

Either the absolute (fully qualified) path to the file you want to add to the design or just the name of the file. If you provide just the file name, the IDE searches the access paths and adds the first file of that name that it finds. If you want to add two one files with identical names, use the fully qualified path to each one.

---

**Designs***FindAndAddFile2*

---

groupPath

The absolute path to the group within which the new file should be added.

projectFile

On return, this parameter contains the address of a pointer to the project file to which the file was added.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

---

**FindAndAddFile2**

This method finds a file and adds it to the design. This method also lets you set link flags on the file.

```
virtual HRESULT FindAndAddFile2(  
    BSTR path,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

Either the absolute (fully qualified) path to the file you want to add to the design or just the name of the file. If you provide just the file name, the IDE searches the access paths and adds the first file of that name that it finds. If you want to add two one files with identical names, use the fully qualified path to each one.

groupPath

The absolute path to the group within which the new file should be added.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

---



projectFile

On return, this parameter contains the address of a pointer to the project file to which the file was added.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

---

## get\_BrowserDB

This method gets a pointer to a listing of the symbols created by a compilation of the files in the design.

```
virtual HRESULT get_BrowserDB(
    ICodeWarriorSymbolContainer **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to a container holding the symbols created by the latest compilation of the files in the design.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSymbolContainer” on page 278](#)

---

## get\_DataModel

This method gets the data model for this design.

```
virtual HRESULT get_DataModel(
    IUnknown **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the data model for the design.

---

## Designs

### *get\_Name*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### get\_Name

This method gets the name of a design.

```
virtual HRESULT get_Name(
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of the design.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### get\_Project

This method gets the project object for the project to which the current design belongs.

```
virtual HRESULT get_Project(
    ICodeWarriorProject **pval) = 0;
```

pval

Upon return, this parameter contains the address of a pointer to the project that the design belongs to.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

---

## get\_Targets

This method gets a list of build targets in this design.

```
virtual HRESULT get_Targets(  
    ICodeWarriorTargetCollection **pval) = 0;
```

pval

Upon return, this parameter contains a collection of the build targets in this design.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## put\_Name

This method to set the name of a design.

```
virtual HRESULT put_Name(  
    BSTR pval) = 0;
```

pval

The name you are assigning to the design.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## RemoveAttachment

This method removes an attachment from the design.

```
virtual HRESULT RemoveAttachment(  
    const CLSID *attachmentCLSID) = 0;
```

## Designs

### *RemoveTargetFromDesign*

---

\*attachmentCLSID

A pointer to the attachment you want to remove.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“AddAttachment” on page 130](#)

---

### RemoveTargetFromDesign

This method removes a target from a design

```
virtual HRESULT RemoveTargetFromDesign(  
    ICodeWarriorTarget *target) = 0;
```

\*target

A pointer to the target you want to remove.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorDesignAttachment

This interface provides methods to detect whether certain events performed on a design have completed.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">DesignClosing</a>	<a href="#">RemovingAttachment</a>
<a href="#">DesignInitialized</a>	

---

### DesignClosing

This method closes the design.

```
virtual HRESULT DesignClosing(  
    ICodeWarriorDesign *__MIDL_0015) = 0;  
  
__MIDL_0015
```

A pointer to the design to be closed.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### DesignInitialized

This method prepares a design for use.

```
virtual HRESULT DesignInitialized(  
    ICodeWarriorDesign *__MIDL_0014) = 0;  
  
__MIDL_0014
```

A pointer to the design you want to initialize.

## Designs

### *RemovingAttachment*

---

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## RemovingAttachment

Use this method to determine if the CodeWarrior IDE has finished removing an attachment from the design.

```
virtual HRESULT RemovingAttachment(  
    ICodeWarriorDesign *__MIDL_0016) = 0;
```

`__MIDL_0016`

A pointer to the attachment being removed.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorDesignEvents

Use this interface to determine if certain events have taken place while working with a design.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

---

[RemovingTarget](#)[TargetAdded](#)

---

---

### RemovingTarget

This method detects whether the CodeWarrior IDE has finished removing a target. (How does one know the target has been removed?)

```
virtual HRESULT RemovingTarget(  
    ICodeWarriorTarget *target) = 0;
```

target

A pointer to the target being removed.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Designs***TargetAdded*

---

---

**TargetAdded**

This method detects whether the CodeWarrior IDE has finished adding a target to the design. (How does one know the target has been added?)

```
virtual HRESULT TargetAdded(  
    ICodeWarriorTarget *target) = 0;
```

target

A pointer to the target being added.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



---

## Data Types

### ECodeWarriorLinkFlags

This enumeration is used to define linker flags. It is used in the [AddFile2](#), [AddFile2ByFileSpec](#), and [FindAndAddFile2](#) methods of the [ICodeWarriorDesign](#) interface.

Constant	Description
<code>cwNoLinkFlags</code>	Set no link flags on this file
<code>cwGenerateSymbols</code>	Instruct the linker to only generate symbols
<code>cwMergeLibrary</code>	Instruct the linker to link to libraries when building.
<code>cwWeakImport</code>	Instruct the linker to use the Weak Import option. This option only works on the Mac OS
<code>cwInitBefore</code>	Instruct the linker to use shared libraries. This flag is only valid if the currently mapped compiler support import. This option works only on the Power PC



# Dialog Services

---

This chapter shows how to use the Dialog Services API to manage dialog operations in the CodeWarrior IDE.

This chapter contains the following sections:

- [Dialog Services API Overview](#)
- [Using the Dialog Services API](#)
- [Dialog Services API Reference](#)

## Dialog Services API Overview

The Dialog Services API is a set of interfaces that allows a plug-in to create and manipulate dialog boxes in the CodeWarrior IDE.

## Using the Dialog Services API

This section covers these topics:

- [Registering the Command](#)
- [Implementing the Command](#)

### Registering the Command

The following code snippet, from `PluginMain.cpp`, shows how to register a dialog command:

---

```
/*
 * RegisterCommands
 *
 * Creates a new command group, adds two commands to it, and
 * tells the IDE to display the group in the menu bar.
 */
```

---

## Dialog Services

### Registering the Command

---

```

static void RegisterCommands(IServiceProvider *servProv)
{
    ICodeWarriorCommandRegistry *cmdRegistry;
    servProv->QueryService(SID_SCodeWarriorCommandRegistry,
        IID_ICodeWarriorCommandRegistry, &cmdRegistry);
    if(cmdRegistry)
    {
        // We don't need the window manager to register commands,
        // but the command handler we're installing will need a
        // reference to it in order to create windows...
        ICodeWarriorWindowManager *windowMgr = NULL;
        servProv->QueryService(SID_SCodeWarriorWindowManager,
            IID_ICodeWarriorWindowManager, &windowMgr);

        ICodeWarriorCommandHandler *cmdHandler = new
            ExampleCommandHandler(windowMgr, servProv);
        BSTR bstr = SysAllocString(OLESTR("Example plugin"));
        cmdRegistry->CreateNewCommandGroup(kToolbarTestPluginID,
            cmdGroup_TestPlugin, bstr, cmdGroup_Nothing);

        CWToolbarIconInfo tbIconInfo =
            GetToolbarIcon(iconIndex_NewPluginWindow);

        // register test dialog services commands
        SysReAllocString(&bstr, OLESTR("Test Info Dialog"));
        cmdInfo.pluginID = kToolbarTestPluginID;
        cmdInfo.commandID = cmd_TestInfoDialog;
        cmdInfo.commandGroupID = cmdGroup_TestPlugin;
        cmdInfo.commandName = bstr;
        cmdInfo.toolbarIcon = tbIconInfo;
        cmdInfo.visibleInMenu = true;
        cmdInfo.itemType = CWCommandItemType_Command;
        cmdInfo.extraInfo.commandHandler = cmdHandler;
        cmdRegistry->RegisterCommand(cmdInfo, cmd_Nothing);

        SysFreeString(bstr);
        FreeToolbarIcon(tbIconInfo);

        cmdRegistry->Release();
    }
}

```

---

```

ICodeWarriorMenuManager *menuMgr;
servProv->QueryService(SID_SCodeWarriorMenuManager,
    IID_ICodeWarriorMenuManager, &menuMgr);
if(menuMgr)
{
    menuMgr->ShowCommandGroupMenu(kToolbarTestPluginID,
        cmdGroup_TestPlugin, true);
    menuMgr->Release();
}
}

```

---

## Implementing the Command

The following code snippet, from ExampleCommandHandler.cpp, shows how to implement a dialog command:

---

```

/*
 *ExecuteCommand
 */

HRESULT STDMETHODCALLTYPE ExampleCommandHandler::ExecuteCommand(
    CWCommandID inCommandNumber,
    ICodeWarriorCommandHandler *inDefaultHandler)
{
    HRESULT result = S_OK;

    switch(inCommandNumber)
    {
        case cmd_TestInfoDialog:
        case cmd_TestWarningDialog:
        case cmd_TestErrorDialog:
        {
            if (mServProv)
            {
                ICodeWarriorDialogServices* dlgSrvc = NULL;
                result =
                    mServProv->QueryService(IID_ICodeWarriorDialogServices,
                        IID_ICodeWarriorDialogServices, (void**) &dlgSrvc);
                if (SUCCEEDED(result))
                {
                    short dialogType = 0;

```

---

## Dialog Services

### Implementing the Command

---

```

    BSTR bstr;
    switch(inCommandNumber)
    {
        case cmd_TestInfoDialog:
            dialogType = cwInfoDialog;
            bstr = SysAllocString(OLESTR("Info OKCancelDialog
            - cwInfoDialog"));
            break;
        case cmd_TestWarningDialog:
            dialogType = cwWarningDialog;
            bstr = SysAllocString(OLESTR("Warning OKCancelDialog
            - cwWarningDialog"));
            break;
        case cmd_TestErrorDialog:
            dialogType = cwErrorDialog;
            bstr = SysAllocString(OLESTR("Error OKCancelDialog -
            cwErrorDialog"));
            break;
    };
    result = dlgSrv->OKCancelDialog(dialogType, bstr);
    SysFreeString(bstr);

    dlgSrv->Release();
}
break;
}

default:
    if(inDefaultHandler)
        result = inDefaultHandler->ExecuteCommand(inCommandNumber,
        inDefaultHandler);
    break;
}

return result;
}

```

---

# Dialog Services API Reference

This section describes the functions contained in the following interface:

- [ICodeWarriorDialogServices](#)

## ICodeWarriorDialogServices

This interface allows plug-ins to present dialogs to the user and request feedback from the user.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">OKCancelDialog</a>	<a href="#">ReportErrorFromErrorInfo</a>
<a href="#">OKDialog</a>	<a href="#">SaveDontSaveDialog</a>
<a href="#">NewItemDialog</a>	<a href="#">SetPluginDialogCommandHandler</a>
<a href="#">PostModalDialog</a>	<a href="#">UpdatePluginDialogMenus</a>
<a href="#">PreModalDialog</a>	

## OKCancelDialog

This method shows the user a dialog with an **OK** button and a **Cancel** button.

```
virtual HRESULT OKCancelDialog(
    short dialogType,
    BSTR message) = 0;
```

dialogType

A value within the range defined by the [Dialog Box Types](#) enumeration, indicating the type of dialog box to present to the

end user.

message

The message you want to display in this dialog.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Dialog Box Types” on page 157](#)

---

## OKDialog

This method shows the user a dialog with an **OK** button.

```
virtual HRESULT OKDialog(  
    short dialogType,  
    BSTR message) = 0;
```

dialogType

A value within the range defined by the [Dialog Box Types](#) enumeration, indicating the type of dialog box to present to the end user.

message

The message you want to display in this dialog.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Dialog Box Types” on page 157](#)

---

## NewItemDialog

This method shows the user a new item dialog.

```
virtual HRESULT NewItemDialog(  
    BSTR pageToSelect,  
    BSTR itemToSelect) = 0;
```

---



pageToSelect

If the dialog has multiple pages, the page to show to the user.  
This parameter is optional

itemToSelect

The item on the selected page to highlight when the dialog appears to the user. If the dialog has multiple pages, the items is relative to the page. This parameter is optional

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## PostModalDialog

This method informs the CodeWarrior IDE that you are done with a modal dialog.

```
PostModalDialog(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## PreModalDialog

This method informs the CodeWarrior IDE that you are about to display a modal dialog. You can then use [SetPluginDialogCommandHandler](#) and [UpdatePluginDialogMenus](#) to update the menu while the modal dialog has focus.

```
virtual HRESULT PreModalDialog(  
    BOOL fActivateIDE) = 0;
```

fActivateIDE

true to bring the IDE to the front or false to leave it behind other applications (if any).

## Dialog Services

### ReportErrorFromErrorInfo

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“SetPluginDialogCommandHandler” on page 155](#)

[“UpdatePluginDialogMenus” on page 156](#)

---

### ReportErrorFromErrorInfo

This method shows the user an error message in a dialog.

```
virtual HRESULT ReportErrorFromErrorInfo(
    ICodeWarriorErrorInfo *info) = 0;
```

info

A pointer to the [ICodeWarriorErrorInfo](#) interface containing the error message to show.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorErrorInfo” on page 176](#)

---

### SaveDontSaveDialog

This method shows a message to the user in a dialog with a **Save** button and a **Don't Save** button..

```
virtual HRESULT SaveDontSaveDialog(
    BSTR objectType,
    BSTR objectName,
    long *result) = 0;
```

objectType

The type of object being saved. If the object is a file, you may pass NULL for this parameter.

---

objectName

The default name of the object being saved. This name appears in the edit field of the dialog box, and may be edited by the user before saving.

result

On return, this parameter contains an integer indicating which option (Save, Don't Save, or Cancel) the user chose.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“SaveDontSaveDialog Result Types” on page 157](#)

---

## SetPluginDialogCommandHandler

This method sets up a command handler to process menu events while a plugin is showing a modal dialog. Call this method after `PreModalDialog()`. After using this method, you can use `UpdatePluginDialogMenus` to update the menu bar while the modal dialog has focus.

```
virtual HRESULT SetPluginDialogCommandHandler(  
    ICodeWarriorCommandHandler* inCommandHandler  
    ) = 0;
```

inCommandHandler

A pointer to the command handler to use while a modal dialog has the focus.

See Also [“PreModalDialog” on page 153](#)

[“UpdatePluginDialogMenus” on page 156](#)

**Dialog Services***UpdatePluginDialogMenus*

---

---

## UpdatePluginDialogMenus

This method updates the menu bar while a modal dialog has the focus. Once a plugin dialog command handler has been set, call this method in your event loop to update the menus. Use this method after calling `SetPluginDialogCommandHandler`.

```
virtual HRESULT UpdatePluginDialogMenus() = 0;
```

See Also [“PreModalDialog” on page 153](#)

[“SetPluginDialogCommandHandler” on page 155](#)

## Dialog Services Data Types




The following data types are used with the Dialog Services API:

- [Dialog Box Types](#)

### Dialog Box Types

These constants are used to describe the types of dialogs available for use with the [OKDialog](#) and [OKCancelDialog](#) methods of the [ICodeWarriorDialogServices](#) interface.

**Table 9.1** Dialog Box Types

Constant	Icon	Description
<code>cwInfoDialog</code>		Used to present information to the end user
<code>cwWarningDialog</code>		Used to present a caution or warning to the end user
<code>cwErrorDialog</code>		Used to present an error condition to the end user

### SaveDontSaveDialog Result Types

The [SaveDontSaveDialog](#) method of the [ICodeWarriorDialogServices](#) interface returns one of three possible values, depending on the user's choice:

- `cwSaveResponse_Save`
- `cwSaveResponse_DontSave`
- `cwSaveResponse_Cancel`



**Dialog Services**  
*Dialog Services Data Types*

---

# Documents

---

This chapter shows how to use the Documents API to create and manage documents in the CodeWarrior IDE.

This chapter contains the following sections:

- [Documents API Overview](#)
- [Documents API Reference](#)

## Documents API Overview

The Documents API is a set of interfaces that allows a plug-in to create and manipulate components in the CodeWarrior IDE.

## Documents API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorDocument](#)
- [ICodeWarriorProjectDocument](#)
- [ICodeWarriorTextDocument](#)

## ICodeWarriorDocument

This interface is used to get information about and control CodeWarrior documents and their windows.

### Inherited Interfaces

- IUnknown

### Methods

This interface has the following properties:

<a href="#">Activate</a>	<a href="#">get_Width</a>
<a href="#">Close</a>	<a href="#">get_XPos</a>
<a href="#">get_ActiveDocument</a>	<a href="#">get_YPos</a>
<a href="#">get_Dirty</a>	<a href="#">put_Height</a>
<a href="#">get_FileSpec</a>	<a href="#">put_Visible</a>
<a href="#">get_Height</a>	<a href="#">put_Width</a>
<a href="#">get_Name</a>	<a href="#">put_XPos</a>
<a href="#">get_ReadOnly</a>	<a href="#">put_YPos</a>
<a href="#">get_Visible</a>	<a href="#">Save</a>

---

### Activate

This method activates a document. Activating a document makes the document the frontmost document.

```
virtual HRESULT Activate(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



## Close

This method closes the document window and optionally saves the document.

```
virtual HRESULT Close(  
    VARIANT_BOOL bSaveChanges) = 0;
```

bSaveChanges

A `VARIANT_BOOL` containing `true` to save the document before closing or `false` to close the document without saving.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## get\_ActiveDocument

This method gets whether a document is the active document.

```
virtual HRESULT get_ActiveDocument(  
    VARIANT_BOOL *pval) = 0;
```

pval

On return it contains `true` if the document is active or `false` if the document is inactive.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## get\_Dirty

This method gets the dirty status (whether or not it has been modified since the last save) of a document.

```
virtual HRESULT get_Dirty(  
    VARIANT_BOOL *pval) = 0;
```

## Documents

### *get\_FileSpec*

---

pval

Supply a pointer to the `VARIANT_BOOL` interface. Upon return it contains `true` if the document is dirty (needs to be written to disk), and `false` if the document is not dirty.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *get\_FileSpec*

This method gets the file specification for a document.

```
virtual HRESULT get_FileSpec(
    IFileSpec **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the file specification for the document.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *get\_Height*

This method gets the height of a document window.

```
virtual HRESULT get_Height(
    int *pval) = 0;
```

pval

On return, this parameter contains a pointer to the current height (in pixels) of the document window.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

---

## get\_Name

This method gets the name of a document.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains a pointer to the name of the document.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_ReadOnly

This method gets the read-only status of a document.

```
virtual HRESULT get_ReadOnly(  
    VARIANT_BOOL *pval) = 0;
```

pval

On return it contains `true` if the document is read only or `false` if the document is modifiable.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Visible

This method gets the visible status of a document window.

```
virtual HRESULT get_Visible(  
    VARIANT_BOOL *pval) = 0;
```

## Documents

### *get\_Width*

---

`pval`

On return, this parameter contains a pointer to a boolean indicating `true` if the document window is visible or `false` if the document window is not visible.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *get\_Width*

This method gets the width of a document window.

```
virtual HRESULT get_Width(
    int *pval) = 0;
```

`pval`

On return, this parameter contains a pointer to an integer indicating the current width (in pixels) of the document window.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *get\_XPos*

This method gets the horizontal position of a document window.

```
virtual HRESULT get_XPos(
    int *pval) = 0;
```

`pval`

On return, this parameter contains a pointer to an integer indicating the current horizontal coordinate of the top-left corner of the document window.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

---

## get\_YPos

This method gets the vertical position of a document window.

```
virtual HRESULT get_YPos(  
    int *pval) = 0;
```

pval

On return, this parameter contains a pointer to an integer indicating the current verticle coordinate of the top-left corner of the document window.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Height

This method sets the height of a document window.

```
virtual HRESULT put_Height(  
    int val) = 0;
```

val

An integer set to the new height (in pixels) of the document window.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Visible

This method sets the visible state of a document window.

```
virtual HRESULT put_Visible(  
    VARIANT_BOOL val) = 0;
```

## Documents

### *put\_Width*

---

val

A `VARIANT_BOOL` set to `true` if you want the document window made visible or `false` if you want the document window hidden.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *put\_Width*

This method sets the width of a document window.

```
virtual HRESULT put_Width(
    int val) = 0;
```

val

An integer set to the new width (in pixels) of the document window.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### *put\_XPos*

This method sets the horizontal position of a document window.

```
virtual HRESULT put_XPos(
    int val) = 0;
```

val

An integer variable set to the new horizontal coordinate of the top-left corner of the document window.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

---

## put\_YPos

This method sets the verticle position of a document window.

```
virtual HRESULT put_YPos(  
    int val) = 0;
```

val

An integer variable set to the new verticle coordinate of the top-left corner of the document window.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Save

This method saves a document.

```
virtual HRESULT Save(void) = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorProjectDocument

This interface is used to get information about and control CodeWarrior project documents.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CollapseGroup</a>	<a href="#">SelectFiles</a>
<a href="#">ExpandGroup</a>	<a href="#">SelectedFiles</a>
<a href="#">get_Project</a>	

---

## CollapseGroup

This method collapses a group in a project window.

```
virtual HRESULT CollapseGroup(
    BSTR groupName) = 0;
```

groupName

The name of the group to collapse.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.



---

## ExpandGroup

This method expands a group in a project window.

```
virtual HRESULT ExpandGroup(  
    BSTR groupName) = 0;
```

groupName

The name of the group to expand.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Project

This method gets the project object related to this document.

```
virtual HRESULT get_Project(  
    ICodeWarriorProject **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the project object related to this document.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorProject" on page 216](#)

## SelectFiles

This method selects or deselects one or more files in a project.

```
virtual HRESULT SelectFiles(
    ICodeWarriorProjectFileCollection
        *projectFiles,
    VARIANT_BOOL select) = 0;
```

projectFiles

A pointer to the ICodeWarriorProjectFileCollection interface containing the list of project files to select.

select

Set this parameter to true if you want to select the files or false if you want to deselect the files.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## SelectedFiles

This method gets a list of currently selected files in a project.

```
virtual HRESULT SelectedFiles(
    ICodeWarriorProjectFileCollection
        **projectFiles) = 0;
```

projectFiles

Supply the address of a pointer to the ICodeWarriorProjectFileCollection interface. Upon return it contains a list of the currently selected project files.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## ICodeWarriorTextDocument

This interface is used to get information about and control CodeWarrior text documents.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">get_TextEngine</a>	<a href="#">SaveAs</a>
<a href="#">SaveACopyAs</a>	<a href="#">SaveAsByFileSpec</a>
<a href="#">SaveACopyAsByFileSpec</a>	<a href="#">ScrollToSelection</a>

---

### get\_TextEngine

This method gets the text engine object of a CodeWarrior text document.

```
virtual HRESULT get_TextEngine(  
    ICodeWarriorTextEngine **pval) = 0;
```

pval

On return, this parameter contains the address of a pointer to the text engine object for this text document.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTextEngine" on page 334](#)

---

## SaveACopyAs

This method saves a text file using the **Save A Copy As** dialog box, by specifying the full path of the file.

```
virtual HRESULT SaveACopyAs(  
    BSTR val) = 0;
```

val

The full path of the file being saved.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SaveACopyAsByFileSpec

This method saves a text file using the **Save A Copy As** dialog box, by specifying the file specification record for the file.

```
virtual HRESULT SaveACopyAsByFileSpec(  
    IFileSpec *fileSpec) = 0;
```

fileSpec

A pointer to the `IFileSpec` interface containing the file specification for the file being saved.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## SaveAs

This method saves a text file using the **Save As** dialog box, by specifying the full path of the file.

```
virtual HRESULT SaveAs(  
    BSTR val) = 0;
```

val

The full path of the file being saved.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SaveAsByFileSpec

This method saves a text file using the **Save As** dialog box, by specifying the file specification record for the file.

```
virtual HRESULT SaveAsByFileSpec(  
    IFileSpec *fileSpec) = 0;
```

fileSpec

A pointer to the `IFileSpec` interface containing the file specification for the file being saved.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## ScrollToSelection

This method makes the currently selected text appear in the text document's editor window.

```
virtual HRESULT ScrollToSelection(void) = 0;
```

## Documents

### *ScrollToSelection*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

# Error Info

---

This chapter shows how to use the Error Info API to manage and work with error information.

This chapter contains the following sections:

- [Error Info API Overview](#)
- [Error Info API Reference](#)

## Error Info API Overview

The Error Info API is a set of interfaces that allows a plug-in to work with error information in the CodeWarrior IDE.

## Error Info API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorErrorInfo](#)

## ICodeWarriorErrorInfo

### Inherited Interfaces

- IUnknown

### Methods

This interface has the following methods:

<a href="#">get_Action</a>	<a href="#">put_HelpContext</a>
<a href="#">get_DWORDErr</a>	<a href="#">put_HelpFile</a>
<a href="#">get_HRESULT</a>	<a href="#">put_HRESULT</a>
<a href="#">get_MacOSErr</a>	<a href="#">put_MacOSErr</a>
<a href="#">get_MWErr</a>	<a href="#">put_MWErr</a>
<a href="#">get_Reason</a>	<a href="#">put_Reason</a>
<a href="#">put_Action</a>	<a href="#">put_Source</a>
<a href="#">put_DWORDErr</a>	

---

### get\_Action

This method gets the action string associated with the most recent error.

```
virtual HRESULT get_Action(BSTR *actionStr) = 0;

actionStr
```

On return, this parameter contains the action string.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



---

## get\_DWORDErr

This method gets the error number of the most recent error.

```
virtual HRESULT get_DWORDErr(DWORD *err) = 0;
```

err

On return, this parameter contains a pointer to a DWORD that contains the error number.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_HRESULT

This method gets the HRESULT value for the most recent operation.

```
virtual HRESULT get_HRESULT(HRESULT *err) = 0;
```

err

On return, this parameter contains a pointer to an HRESULT that contains the HRESULT value for the most recent operation (0 for no error and other values for errors).

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_MWErr

This method gets the MWErr number for the most recent error.

```
virtual HRESULT get_MWErr(long *err) = 0;
```

err

On return, this parameter contains a pointer to a long integer

**Error Info**  
*get\_MacOSErr*

---

containing the error number

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**get\_MacOSErr**

This method gets the error number of the most recent error on a Mac OS.

```
virtual HRESULT get_MacOSErr(short *err) = 0;
```

err

On return, this parameter contains a pointer to a short that holds the error number.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**get\_Reason**

This method gets a string that contains an explanation of what caused the most recent error.

```
virtual HRESULT get_Reason(BSTR *actionStr) = 0;
```

actionStr

On return, this parameter contains a string containing the cause of the error.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Action

This method lets you set the action string for an error message.

```
virtual HRESULT put_Action(BSTR actionStr) = 0;  
actionStr
```

A string telling the user what action to take to clear the error.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Error Info

*put\_DWORDErr*

---

### put\_DWORDErr

This method lets you set the error number for an error.

```
virtual HRESULT put_DWORDErr(DWORD err) = 0;
```

err

The number you want to assign to the error.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### put\_HRESULT

This method lets you set the HRESULT value for an error.

```
virtual HRESULT put_HRESULT(HRESULT err) = 0;
```

err

The HRESULT value you want to assign to an error.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### put\_HelpContext

This method lets you put the help system into a certain state before displaying a help file. Refer to the operating system documentation for available states for the help system.

```
virtual HRESULT put_HelpContext(
    DWORD helpContext) = 0;
```

helpContext

A number indicating the state the help system should be in

---

when you display a help file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_HelpFile

This method lets you show the user a help file. You can use [put\\_HelpContext](#) to put the help system into a particular state before you call the help file.

```
virtual HRESULT put_HelpFile(BSTR fileName) = 0;
```

fileName

The file name (and path, if necessary) of the help system to show the user.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_MWErr

This method lets you set the MWErr value for an error message.

```
virtual HRESULT put_MWErr(long err) = 0;
```

err

A number representing an MWErr state.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Error Info

*put\_MacOSErr*

---

### put\_MacOSErr

---

This method lets you set the error number to a value recognized by the Mac OS.

```
virtual HRESULT put_MacOSErr(short err) = 0;
```

err

The Mac OS error number to use.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Reason

This method lets you set the reason string for an error message.

```
virtual HRESULT put_Reason(BSTR actionStr) = 0;  
  
actionStr
```

A string indicating the reason the error occurred.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## put\_Source

This method lets you set the source of the error message.

```
virtual HRESULT put_Source(BSTR sourceStr) = 0;  
  
sourceStr
```

A string stating the source of the error.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.



**Error Info**  
*put\_Source*

---



# Files

---

This chapter describes the functions contained in the following interface: [IFileSpec](#)

## IFileSpec

This interface allows you to work with file specifications.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">Clone</a>	<a href="#">get_Name</a>
<a href="#">Copy</a>	<a href="#">put_FullPath</a>
<a href="#">get_FullPath</a>	<a href="#">put_Name</a>

---

## Clone

This method creates a duplicate copy of a file specification pointing to the same file as the original.

```
virtual HRESULT Clone(IFileSpec **pval);
```

pval

On return, this parameter contains the address of a pointer to an `IFileSpec` object pointing to the same file as the original file specification.

## Files

### Copy

---

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## Copy

This method creates a duplicate copy of a file, by specifying the name and location of the new file.

```
virtual HRESULT Copy(IFileSpec *inSpec);
```

inSpec

A pointer to the `IFileSpec` interface containing a file specification with the name and the location of the new file.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## get\_FullPath

This method gets the full path of a file specification.

```
virtual HRESULT get_FullPath(BSTR *path);
```

pval

On return, this parameter contains the full path of the file.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## get\_Name

This method gets the name of a file specification.

```
virtual HRESULT get_Name(BSTR *pval);
```

---

pval

On return, this parameter contains the name of the file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_FullPath

This method sets the path of a file specification.

```
virtual HRESULT put_FullPath(BSTR path);
```

pval

The new path for the file specification. The path you supply does not need to point to an existing file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_Name

This method sets the name of a file specification.

```
virtual HRESULT put_Name(BSTR pval);
```

pval

The new name of the file. The name you supply does not have to point to an existing file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---



**Files**

*put\_Name*

---

# Menus

---

This chapter shows how to use the Menu API to create and manage menus in the CodeWarrior IDE.

This chapter contains the following sections:

- [Menu API Overview](#)
- [Using the Menu API](#)
- [Menu API Reference](#)

## Menu API Overview

The Menu API is a set of interfaces that allows a plug-in to create and manipulate menus in the CodeWarrior IDE. The API uses the standard COM interface.

Menu can either be created at IDE initialization or during run time. Menu should be created at IDE launch via the `ICodeWarriorMenuManager` interface while dynamic menus should be created with the `ICodeWarriorMenu` interface.

Run time manipulation of menus (`ICodeWarriorMenu` and `ICodeWarriorMenuHandler`) is not yet implemented.

## Using the Menu API

The header file, `CodeWarriorMenuManager.h` defines all the interfaces for the menu API. To create a menu at launch you will need to create a command handler. See [ICodeWarriorCommandRegistry](#) for more information on creating a command group at IDE launch.

An example of creating a menu at launch is shown in [Listing 13.1](#).

## Menus

*Menus API Reference*

---

### Listing 13.1 Creating a menu at launch

---

```
ICodeWarriorMenuManager *menuMgr;

servProv->QueryService(SID_SCodeWarriorMenuManager,
    IID_ICodeWarriorMenuManager, &menuMgr);
if(menuMgr)
{
    menuMgr->ShowCommandGroupMenu(kToolbarTestPluginID,
        cmdGroup_TestPlugin, true);
    menuMgr->Release();
}
```

---

## Menus API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorMenu](#)
- [ICodeWarriorMenuHandler](#)
- [ICodeWarriorMenuManager](#)

## ICodeWarriorMenu

This interface works with the entire menu bar, a single menu in the menu bar, or a single menu item. The methods in this interface allow you to manage menus and menu items at run time. The interface grants no access to key bindings, toolbars, or submenus.

### Inherited Interfaces

- IUnknown

---

**NOTE** ICodeWarriorMenu methods work only with ICodeWarriorMenu interfaces. There is no mechanism for adding or removing an item from a built-in menu at runtime. You can add items to existing menus when CodeWarrior launches using the command registry mechanism. See [“Commands API Overview” on page 83](#) for more information on using the command registry mechanism for creating menus.

---

### Methods

This interface provides the following methods:

<a href="#">InsertItem</a>	<a href="#">SetItemChecked</a>
<a href="#">RemoveAllItems</a>	<a href="#">SetItemEnabled</a>
<a href="#">RemoveItem</a>	<a href="#">SetItemName</a>

---

---

## InsertItem

**This method inserts a menu item on a menu.**

```
virtual HRESULT InsertItem(  
    BSTR inItemName,  
    LONG inBeforeIndex);
```

`inItemName`

The name of the item you wish to appear in the menu.

`inBeforeIndex`

The location of the menu item you wish to insert. The new menu is placed before the value specified in `inBeforeIndex`.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## RemoveAllItems

**This method removes all items from a menu.**

```
virtual HRESULT RemoveAllItems(void);
```

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.



---

## RemoveItem

This method removes a single item from a menu.

```
virtual HRESULT RemoveItem(LONG inItemIndex);
```

`inItemIndex`

Specifies the item index number for the menu item you want to remove.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## SetItemChecked

This method shows or hides a checkmark next to a menu item.

```
virtual HRESULT SetItemChecked(  
    LONG inItemIndex,  
    BOOL inNewState);
```

`inItemIndex`

The menu item that is selected.

`inNewState`

Set this parameter to `true` for a check mark next to the menu item or `false` for no checkmark.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Menus***SetItemEnabled*

---

---

**SetItemEnabled**

Enables or disables a specified menu item.

```
virtual HRESULT SetItemEnabled(  
    LONG inItemIndex,  
    BOOL inNewState);
```

*inItemIndex*

This is a user-defined command ID which is assigned to this menu item when it is created. The command ID must already be registered with the IDE via [RegisterCommand](#).

*inNewState*

Specifies whether the menu item is enabled or not. Set *inNewState* to `true` to enable the specified menu item or `false` to disable the menu item.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [RegisterCommand](#)

---

**SetItemName**

This method changes the name for a menu item.

```
virtual HRESULT SetItemName(  
    LONG inItemIndex,  
    BSTR inNewName);
```

*inItemIndex*

Specifies the index for the menu item whose name you want to change.

*inNewName*

The name of the menu item.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorMenuHandler

This interface handles events and status for menu item plug-ins. To use it, you must create a COM object that inherits from this interface.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">HandleMenuSelection</a>	<a href="#">UpdateMenuStatus</a>
-------------------------------------	----------------------------------

---

## HandleMenuSelection

The IDE calls this method to let the plug-in know to perform the action associated with the selected menu item.

```
virtual HRESULT HandleMenuSelection(  
    long inItemIndex,  
    BSTR inItemName);
```

`inItemIndex`

The item number of the menu item whose event you want to dispatch.

`inItemName`

The name of the menu item.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## UpdateMenuStatus

This method updates a menu item.

```
virtual HRESULT UpdateMenuStatus(void);
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorMenuManager

This interface allows you to obtain the interface for the CodeWarrior menu bar.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">CreateTemporaryMenu</a>	<a href="#">SetMenusEnabledState</a>
<a href="#">GetMenusEnabledState</a>	<a href="#">ShowCommandGroupMenu</a>

---

## CreateTemporaryMenu

This method creates a temporary menu.

```
virtual HRESULT CreateTemporaryMenu(
    BSTR inMenuTitle,
    ICodeWarriorMenuHandler *inHandler,
    ICodeWarriorMenu *&outMenuInterface) = 0;
```

*inMenuTitle*

The title of the menu you want to create.

*inHandler*

A pointer to the [ICodeWarriorMenuHandler](#) object you are referring to.

*outMenuInterface*

A reference to a pointer of the [ICodeWarriorMenu](#) object.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorMenuHandler" on page 196](#)

["ICodeWarriorMenu" on page 191](#)

---

## GetMenusEnabledState

This method gets whether the menu bar is enabled or disabled.

```
virtual BOOL GetMenusEnabledState() = 0;
```

Returns **true** if the menu bar is enabled or **false** if not.

---

## SetMenusEnabledState

This method enables or disables the menu bar.

```
virtual HRESULT SetMenusEnabledState(  
    BOOL inEnableMenus) = 0;
```

`inEnabledMenus`

Set this parameter to **true** to enable the menu bar or **false** to disable it.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Menus***ShowCommandGroupMenu*

---

---

**ShowCommandGroupMenu**

This method shows the menu you have created through the methods in the [ICodeWarriorMenu](#) interface. To make the menu appear, you first have to register it, by calling the `IServiceProvider::QueryInterface()` method with the [ICodeWarriorMenuManager](#) object.

```
virtual HRESULT ShowCommandGroupMenu(  
    CWPluginID inPluginID,  
    CWCommandGroupID inCommandGroup,  
    BOOL inShow) = 0;
```

`inPluginID`

The ID number of the plug-in you are creating.

`inCommandGroup`

The menu group in which your menu items are to receive commands. You must create a command group first.

`inShow`

Set this parameter to `true` to show the menu or `false` to hide the menu item referred to in the `inCommandGroup` parameter.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“CreateNewCommandGroup” on page 92](#)



# Messages

---

This chapter shows how to use the Messages API to create and manage messages in the CodeWarrior IDE.

This chapter contains the following sections:

- [Messages API Overview](#)
- [Messages API Reference](#)

## Messages API Overview

The Messages API is a set of interfaces that allows a plug-in to create and manipulate access paths in the CodeWarrior IDE.

## Messages API Reference

This section describes the methods contained in the following interfaces:

- [ICodeWarriorBuildMessages](#)
- [ICodeWarriorMessage](#)

It also describes a data type used by the Message API:

- [Message Data Types](#)

## ICodeWarriorBuildMessages

This interface allows you to examine errors generated by the CodeWarrior IDE while it builds a project.

### Inherited Interfaces

- IUnknown

### Properties

This interface has the following properties:

<a href="#">get_DefinitionCount</a>	<a href="#">get_InformationCount</a>
<a href="#">get_Definitions</a>	<a href="#">get_Information</a>
<a href="#">get_ErrorCount</a>	<a href="#">get_WarningCount</a>
<a href="#">get_Errors</a>	<a href="#">get_Warnings</a>

---

### get\_DefinitionCount

This method gets the number of definitions generated during the last build.

```
virtual HRESULT get_DefinitionCount(
    long *count) = 0;
```

count

On return, this parameter contains a pointer to the number of definitions generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Definitions

This method gets a collection of definitions generated during the last build.

```
virtual HRESULT get_Definitions(  
    ICodeWarriorMessageCollection **errors) = 0;
```

errors

On return, this parameter contains a collection of all definitions generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_ErrorCount

Use this method to obtain the number of errors generated during the last build.

```
virtual HRESULT get_ErrorCount(  
    long *count) = 0;
```

count

Supply a pointer to a long. Upon return it contains the number of errors generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Messages***get\_Errors*

---

**get\_Errors**

This method gets a collection of errors generated during the last build.

```
virtual HRESULT get_Errors(  
    ICodeWarriorMessageCollection **errors) = 0;
```

errors

On return, this parameter contains the address of a pointer to a collection of all errors generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["Using the Collections API" on page 77](#)

---

**get\_InformationCount**

This method gets the number of information items generated during the last build.

```
virtual HRESULT get_InformationCount(  
    long *count) = 0;
```

count

Supply a pointer to a long. Upon return it contains the number of errors generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Information

This method gets a collection of information items generated during the last build.

```
virtual HRESULT get_Information(  
    ICodeWarriorMessageCollection **info) = 0;
```

info

On return, this parameter contains the address of a pointer to a collection of all information items generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_WarningCount

This method gets the number of warnings generated during the last build.

```
virtual HRESULT get_WarningCount(long *count) = 0;
```

count

On return, this parameter contains a pointer to the number of warnings generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Messages***get\_Warnings*

---

---

**get\_Warnings**

Use this method to obtain a collection of warnings generated during the last build.

```
virtual HRESULT get_Warnings(  
    ICodeWarriorMessageCollection **warnings) = 0;
```

warnings

On return, this parameter contains the address of a pointer to a collection of all warnings generated during the last build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## ICodeWarriorMessage

This interface defines a message used by the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Properties

This interface contains the following properties:

<a href="#">get_ErrorNumber</a>	<a href="#">get_SourceLineNumber</a>
<a href="#">get_FileSpec</a>	<a href="#">get_SourceOffset</a>
<a href="#">get_MessageLength</a>	<a href="#">get_Target</a>
<a href="#">get_MessageLineCount</a>	<a href="#">get_TokenLength</a>
<a href="#">get_MessageText</a>	<a href="#">get_TokenOffset</a>
<a href="#">get_ProjectFile</a>	<a href="#">get_Type</a>
<a href="#">get_SourceLength</a>	

---

### get\_ErrorNumber

This method gets the error number of the most recent error.

```
virtual HRESULT get_ErrorNumber(
    long *errorNumber) = 0;
```

errorNumber

On return, this parameter contains a pointer to a long integer that is the error number.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Messages***get\_FileSpec*

---

---

**get\_FileSpec**

This method gets the file specification associated with the current message.

```
virtual HRESULT get_FileSpec(  
    IFileSpec **fileSpec) = 0;
```

fileSpec

On return, this parameter contains the address of a pointer to the file specification of the message.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

**get\_MessageLength**

This method gets the length of the current message, in bytes.

```
virtual HRESULT get_MessageLength(  
    long *messageLength) = 0;
```

messageLength

On return, this parameter contains a pointer to a long integer that holds the length of the message, in bytes.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



---

## get\_MessageLineCount

This method gets the number of lines in the current message.

```
virtual HRESULT get_MessageLineCount(  
    long *lineCount) = 0;
```

lineCount

On return, this parameter contains a pointer to a long integer that holds the number of lines in the message.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_MessageText

This message returns a string containing the text of the current message.

```
virtual HRESULT get_MessageText(  
    BSTR *message) = 0;
```

message

A string containing the text of the current message.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## Messages

*get\_ProjectFile*

---

### get\_ProjectFile

This method gets the project file of the project associated with the current message.

```
virtual HRESULT get_ProjectFile(
    ICodeWarriorProjectFile **projectFile) = 0;
```

projectFile

On return, this parameter contains the address of a pointer to the project file object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

---

### get\_SourceLength

This method gets the length of the source file that the project was manipulating when it generated the current message.

```
virtual HRESULT get_SourceLength(
    long *length) = 0;
```

length

On return, this parameter contains a pointer to a long integer containing the length of the source file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_SourceLineNumber

This method gets the line number within the source file where a problem (or other event) caused a message to be created.

```
virtual HRESULT get_SourceLineNumber(  
    long *lineNumber) = 0;
```

lineNumber

On return, this parameter contains a pointer to a long integer containing the line number within the source file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_SourceOffset

This method gets the number of characters, from the beginning of the source file, to the start of the keyword or phrase that caused the message to be created.

```
virtual HRESULT get_SourceOffset(  
    long *offset) = 0;
```

offset

On return, this parameter contains a pointer to a long integer containing the number of characters, from the beginning of the line, to the start of the keyword in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Messages***get\_Target*

---

---

**get\_Target**

This method gets the keyword or phrase that caused the message to be created.

```
virtual HRESULT get_Target(  
    ICodeWarriorTarget **target) = 0;
```

target

On return, this parameter contains the address of a pointer to the keyword or phrase that caused the message to be created.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTarget” on page 286](#)

---

**get\_TokenLength**

This method gets the length (the number of characters) of the keyword or phrase that caused the message to be created.

```
virtual HRESULT get_TokenLength(  
    long *tokenLength) = 0;
```

tokenLength

On return, this parameter contains a pointer to a long integer containing the number of characters in the keyword or phrase that caused the message.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_TokenOffset

This method gets the number of characters, from the beginning of the line, to the start of the keyword or phrase that caused the message to be created.

```
virtual HRESULT get_TokenOffset(  
    long *tokenOffset) = 0;
```

tokenOffset

On return, this parameter contains a pointer to a long integer with the number of characters from the beginning of the line to the keyword or phrase that caused a message to be created.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Type

This method gets the type of the message.

```
virtual HRESULT get_Type(  
    EMsgType *type) = 0;
```

type

On return, this parameter contains a pointer to the message type.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EMsgType” on page 214](#)

## Message Data Types

The following enumeration defines the message types used in the CodeWarrior Message API.

### EMsgType

This enumeration is used to define messages created by the build process. It is used by the [get\\_Type](#) method in the [ICodeWarriorMessage](#) interface.

Constant	Description
typeNotDefined	This message type is undefined.
typeInformation	This message is an information item.
typeWarning	This message is a warning.
typeError	This message describes an error.
typeDefinition	This message is a definition.

# Projects

---

This chapter shows how to use the Projects API to create and manage projects using plug-in interfaces.

This chapter contains the following sections:

- [Projects API Overview](#)
- [Projects API Reference](#)

## Projects API Overview

The Projects API is a set of interfaces that allows a plug-in to work with projects in the CodeWarrior IDE.

## Projects API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorProject](#)
- [ICodeWarriorProjectAssociation](#)
- [ICodeWarriorProjectEvents](#)
- [ICodeWarriorProjectFile](#)

These interfaces make use of various data types, which are described in the following section:

- [Project Data Types](#)

## ICodeWarriorProject

This interface defines a CodeWarrior project.

### Inherited Interfaces

- IUnknown

### Methods

This interface provides the following methods:

<a href="#">Build</a>	<a href="#">get_FileSpec</a>
<a href="#">BuildWithOptions</a>	<a href="#">get_IsVisible</a>
<a href="#">BuildAndWaitToComplete</a>	<a href="#">get_Name</a>
<a href="#">BuildAndWaitToCompleteWithOptions</a>	<a href="#">GetNamedPluginData</a>
<a href="#">Close</a>	<a href="#">get_Targets</a>
<a href="#">CloneTarget</a>	<a href="#">get_VersionControl</a>
<a href="#">CompileFilesWithChoice</a>	<a href="#">RemoveDesign</a>
<a href="#">CreateDesign</a>	<a href="#">RemoveDesignByName</a>
<a href="#">CreateTarget</a>	<a href="#">RemoveFile</a>
<a href="#">Export</a>	<a href="#">RemoveNamedPluginData</a>
<a href="#">ExportByFileSpec</a>	<a href="#">RemoveObjectCode</a>
<a href="#">FindDesign</a>	<a href="#">RemoveObjectCodeWithOptions</a>
<a href="#">FindFileByName</a>	<a href="#">RemoveTarget</a>
<a href="#">FindTarget</a>	<a href="#">ReportMessage</a>
<a href="#">get_Application</a>	<a href="#">SetCurrentTarget</a>
<a href="#">GetCurrentTarget</a>	<a href="#">SetNamedPluginData</a>
<a href="#">get_Designs</a>	<a href="#">SynchronizeStatus</a>

---

## Build

This method starts a build of the current project.

```
virtual HRESULT Build(
```



```
long *cookie);
```

cookie

On return, this parameter contains a unique identifier for the build process.

Returns. S\_OK if this method call succeeded or an appropriate error if it failed.

---

## BuildWithOptions

This method builds the current project with one of the options specified in the [ECodeWarriorBuildOptions](#) enumeration.

```
virtual HRESULT BuildWithOptions(  
    ECodeWarriorBuildOptions options,  
    ECodeWarriorRunMode runMode,  
    long *cookie) = 0;
```

options

The build options to use with this build.

runmode

Whether to run the resulting program after building it and, if so, whether to run it in debug mode. The [ECodeWarriorRunMode](#) enumeration contains the constants that define this parameter.

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorBuildOptions” on page 249](#)

[“ECodeWarriorRunMode” on page 249](#)

## BuildAndWaitToComplete

This method starts a build of the current project and has the IDE wait to gather all messages from the build process.

```
virtual HRESULT BuildAndWaitToComplete(
    ICodeWarriorBuildMessages **buildMessages);
```

buildMessages

On return, this parameter contains the address of a pointer to the messages created by the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorBuildMessages” on page 202](#)

---

## BuildAndWaitToCompleteWithOptions

This method builds the current project with one of the options specified in the [ECodeWarriorBuildOptions](#) enumeration. This method accumulates all the messages from the build process before returning.

```
virtual HRESULT BuildAndWaitToCompleteWithOptions(
    ECodeWarriorBuildOptions options,
    ICodeWarriorBuildMessages **buildMessages
) = 0;
```

options

The build options to use with this build.

buildMessages

On return, this parameter contains the address of a pointer to the build messages created by the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorBuildOptions” on page 249](#)  
[“ICodeWarriorBuildMessages” on page 202](#)

---

## CloneTarget

This method puts a copy of a specified target into a new directory.

```
virtual HRESULT CloneTarget(  
    ICodeWarriorTarget *srcTarget,  
    ICodeWarriorProject *srcProject,  
    BSTR inDestTargetName,  
    VARIANT_BOOL fCopyFileList,  
    VARIANT_BOOL fCopyTargetSettings,  
    ICodeWarriorDesign *design,  
    ICodeWarriorTarget **outTarget);
```

srcTarget

A pointer to the target to be cloned.

srcProject

A pointer to the project that contains the target to be cloned.

inDestTargetName

The new name for the cloned target.

fCopyFileList

Set this parameter to true to copy the source target's files or false to create a target with no files.

fCopyTargetSettings

Set this parameter to true to copy the source target's settings or false to create a target with default settings.

design

A pointer to the design associated with the source target.

## Projects

### Close

---

outTarget

On return, this parameter contains the address of a pointer to the new target object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTarget" on page 286](#)

["ICodeWarriorDesign" on page 130](#)

---

### Close

This method closes the current project.

```
virtual HRESULT Close(void);
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### CompileFilesWithChoice

This method compiles a collection of project files with one of the options specified in the [ECodeWarriorCompileChoice](#) enumeration.

```
virtual HRESULT CompileFilesWithChoice(
    ICodeWarriorProjectFileCollection *collection,
    ECodeWarriorCompileChoice compileChoice,
    long *cookie) = 0;
```

collection

A pointer to a collection of file projects to compile.

compileChoice

The kind of compilation the compiler should perform.

---

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorCompileChoice” on page 248](#)

[“Using the Collections API” on page 77](#)

---

## CreateDesign

This method creates a new design within the current project

```
virtual HRESULT CreateDesign(  
    BSTR designName,  
    ICodeWarriorDesign **design);
```

designName

The name for the new design.

design

On return, this parameter contains the address of a pointer to the new design object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorDesign” on page 130](#)

---

## CreateTarget

This method creates a new target within the current project.

```
virtual HRESULT CreateTarget(  
    BSTR targetName,  
    BSTR linkerName,
```

---

## Projects

### Export

---

```

IcodeWarriorDesign *design,
IcodeWarriorTarget **target);

```

targetName

The name of the new target.

linkerName

The name of the linker to use to build the new target.

design

The name of the design to associate with the new target.

target

On return, this parameter contains the address of a pointer to the new target object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Export

This method exports the current project to a new location in the file system.

```
virtual HRESULT Export(BSTR filePath);
```

filePath

The full path of the new location.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ExportByFileSpec

This method exports the current project to a file specification.

```
virtual HRESULT ExportByFileSpec(  
    IFileSpec *fileSpec);
```

fileSpec

The file specification to which to export the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## FindDesign

This method finds a particular design within the project, given the name of the design.

```
virtual HRESULT FindDesign(  
    BSTR name,  
    ICodeWarriorDesign **design);
```

name

The name of the design to find.

design

On return, this parameter contains the address of a pointer to the design specified by the name parameter.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorDesign” on page 130](#)

---

## FindFileByName

This method finds a file within the current project by finding the file's name.

```
virtual HRESULT FindFileByName(  
    BSTR fileName,  
    ICodeWarriorProjectFileCollection  
    **projectFiles) = 0;
```

fileName

The name of the file to find.

projectFiles

On return, this parameter contains the address of a pointer to the file specified in the name parameter.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## FindTarget

This method finds a particular target within the current project.

```
virtual HRESULT FindTarget(  
    BSTR name,  
    ICodeWarriorTarget **target);
```

name

The name of the target to find.

target

On return, this parameter contains the address of a pointer to the target specified by the name parameter.



Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTarget" on page 286](#)

---

## get\_Application

This method gets a pointer to the current pointer's application object.

```
virtual HRESULT get_Application(  
    ICodeWarriorApp **val);
```

val

On return, this parameter contains the address of a pointer to the application object associated with the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorApp" on page 35](#)

---

## GetCurrentTarget

This method gets a pointer to the current target within the current project.

```
virtual HRESULT GetCurrentTarget(  
    ICodeWarriorTarget **target);
```

target

On return, this parameter contains the address of a pointer to the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorTarget" on page 286](#)

---

---

## get\_Designs

This method gets a collection containing the designs within the current project.

```
virtual HRESULT get_Designs(
    ICodeWarriorDesignCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to a collection containing the designs within the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_FileSpec

This method gets the file specification of the current project.

```
virtual HRESULT get_FileSpec(
    IFileSpec **pval);
```

pval

On return, this parameter contains the address of a pointer to the current project’s file specification.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## get\_IsVisible

This method gets whether the current project is visible in the IDE.

```
virtual HRESULT get_IsVisible(  
    VARIANT_BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the project is visible or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Projects

*get\_Name*

---

### get\_Name

This method gets the name of the current project.

```
virtual HRESULT get_Name(BSTR *pval);
```

pval

On return, this parameter contains the name of the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### GetNamedPluginData

This method gets plug-in data for the project, by name.

```
virtual HRESULT GetNamedPluginData(
    BSTR resourceName,
    EPluginDataStorageLoc storeIn,
    IStream **pluginData);
```

resourceName

The name of the plug-in from which to get the data.

storeIn

The location of the plug-in's data storage.

pluginData

On return, this parameter contains the address of a pointer to the plug-in data.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## get\_Targets

This method gets a collection containing the targets within the current project.

```
virtual HRESULT get_Targets(  
    ICodeWarriorTargetCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to a collection of targets.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_VersionControl

This method gets the version control object for the current project.

```
virtual HRESULT get_VersionControl(  
    ICodeWarriorVersionControl **versionControl);
```

versionControl

On return, this parameter contains the address of a pointer to the current project’s version control object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorVersionControl” on page 370](#)

---

## RemoveDesign

This method removes a design (and possibly any designs nested within the specified design) from the project.

```
virtual HRESULT RemoveDesign(
    ICodeWarriorDesign *design,
    VARIANT_BOOL fDeleteContainedDesigns);
```

*design*

A pointer to the design to remove.

*fDeleteContainedDesigns*

---

**NOTE** *fDeleteContainedDesigns* indicates whether to delete targets, not designs.

---

Set this parameter to `true` to delete any targets contained within the design specified in the *design* parameter or `false` to leave the targets.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorDesign” on page 130](#)

---

## RemoveDesignByName

This method removes a design (and possibly any designs nested within the specified design) from the project, given a design name.

```
virtual HRESULT RemoveDesignByName(
    BSTR designName,
    VARIANT_BOOL fDeleteContainedTargets);
```

*designName*

The name of the design to remove.

fDeleteContainedDesigns

**NOTE** fDeleteContainedDesigns indicates whether to delete targets, not designs.

---

Set this parameter to `true` to delete any designs contained within the design specified in the design parameter or `false` to leave the targets.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## RemoveFile

This method removes a specified project file.

```
virtual HRESULT RemoveFile(  
    ICodeWarriorProjectFile *projectFile)  
  
projectFile
```

A pointer to the project file to remove.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

---

## RemoveNamedPluginData

This method removes plug-in data, given a name for the plug-in.

```
virtual HRESULT RemoveNamedPluginData(  
    BSTR resourceName,  
    EPluginDataStorageLoc storeIn);  
  
resourceName
```

The name of the plug-in from which to remove data.

## Projects

### RemoveObjectCode

---

storeIn

The location of the data.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## RemoveObjectCode

This method removes the object code from a specified target.

```
virtual HRESULT RemoveObjectCode(
    ECodeWarriorWhichTargetOptions whichTarget,
    VARIANT_BOOL compact);
```

whichTarget

The target from which to remove object code.

compact

true to have the IDE destroy the associated data files (from the data folder) and re-create them or false to leave the data files as they are.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorWhichTargetOptions” on page 331](#)

---

## RemoveObjectCodeWithOptions

This method removes the object code from a single target or all targets. It also lets you choose whether to remove the object code from all the subprojects within the current project and whether to delete any associated data files.

```
virtual HRESULT RemoveObjectCodeWithOptions(
```

---



```
ECodeWarriorWhichTargetOptions whichTarget,  
VARIANT_BOOL recurseSubProject,  
VARIANT_BOOL deleteDataFiles) = 0;
```

whichTarget

A value with the range defined by the [ECodeWarriorWhichTargetOptions](#) enumeration that specifies whether to remove the object code from all targets or only the current target.

## Projects

### RemoveTarget

---

`recurseSubProject`

Set this parameter to `true` to remove the object code within all the subprojects that match the `whichTarget` parameter or `false` to leave the object code in the subprojects.

`deleteDataFiles`

Set this parameter to `true` to delete data files associated with the current project or `false` to retain the data files.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorWhichTargetOptions” on page 331](#)

---

## RemoveTarget

This method removes the specified target from the current project.

```
virtual HRESULT RemoveTarget(
    ICodeWarriorTarget *target);
```

`target`

A pointer to the target to remove.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTarget” on page 286](#)

## ReportMessage

This method makes the specified message appear in the IDE's message window.

```
virtual HRESULT ReportMessage(  
    EReportMsgType msgType,  
    BSTR message);
```

msgType

The type of message (information, warning, etc.) to report.

message

The message to report.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Message Data Types” on page 214](#)

---

## SetCurrentTarget

This method sets the current build target within the project.

```
virtual HRESULT SetCurrentTarget(  
    BSTR targetName);
```

targetName

The name of the target to set as the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Projects***SetNamedPluginData*

---

---

## SetNamedPluginData

This method sets the data for a plug-in.

```
virtual HRESULT SetNamedPluginData(  
    BSTR resourceName,  
    EPluginDataStorageLoc storeIn,  
    IStream *pluginData);
```

resourceName

The name of the plug-in.

storeIn

The location in which to store the plug-in's data.

pluginData

The data to store in the plug-in.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## SynchronizeStatus

This method synchronizes all the file dates within the current project.

```
virtual HRESULT SynchronizeStatus(void);
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorProjectAssociation

This interface provides access to the project associated with the current project file.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

---

[get\\_Project](#)[put\\_Project](#)

---

---

### get\_Project

The IDE calls this method to get the project associated with the current project file.

```
virtual HRESULT get_Project(  
    ICodeWarriorProject **pval);
```

pval

On return, this parameter contains the address of a pointer to the project associated with the current project file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

## Projects

*put\_Project*

---

---

### put\_Project

The IDE calls this method to associate a new project object with the current project file.

```
virtual HRESULT put_Project(  
    ICodeWarriorProject *pval);
```

pval

A pointer to the project to associate with the current project file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

## ICodeWarriorProjectEvents

This interface provides a way to create events while a user works with a project.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">BuildEnded</a>	<a href="#">QueryAboutToBuild</a>
<a href="#">BuildStarted</a>	<a href="#">QueryDeleteDesign</a>
<a href="#">DeletingDesign</a>	<a href="#">QueryUIClose</a>
<a href="#">DesignCreated</a>	<a href="#">RevertCompleted</a>
<a href="#">ProjectClosing</a>	<a href="#">VisibleChanged</a>

---

### BuildEnded

This method indicates that a build has ended.

```
virtual HRESULT BuildEnded(  
    ECodeWarriorCompileChoice choice,  
    long buildID,  
    VARIANT_BOOL fBuildSucceeded,  
    ICodeWarriorBuildMessages *buildMessages);
```

choice

The kind of operation the compile performed.

buildID

The ID of the build.

fBuildSucceeded

Set this parameter to `true` if the build succeeded or `false` otherwise.

buildMessages

A pointer to the messages generated by the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorCompileChoice” on page 248](#)  
[“ICodeWarriorBuildMessages” on page 202](#)

---

## BuildStarted

This method indicates that a build has started.

```
virtual HRESULT BuildStarted(
    ECodeWarriorCompileChoice choice,
    long buildID,
    ICodeWarriorTargetCollection *targetList);
```

choice

The kind of operation the compile performed.

buildID

The ID of the build.

targetList

A pointer to the collection of targets to build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorCompileChoice” on page 248](#)  
[“Using the Collections API” on page 77](#)

---



---

## DeletingDesign

This method indicates that a design is being deleted.

```
virtual HRESULT DeletingDesign(  
    ICodeWarriorDesign *design);
```

design

A pointer to the design to delete.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## DesignCreated

This method indicates that a design is being created.

```
virtual HRESULT DesignCreated(  
    ICodeWarriorDesign *design);
```

design

A pointer to the design to create.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## ProjectClosing

This method indicates that a project is being closed.

```
virtual HRESULT ProjectClosing(  
    ICodeWarriorProject *project);
```

project

A pointer to the project to close.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

---

## QueryAboutToBuild

The CodeWarrior IDE calls this method to inform the plug-in that a build is about to start.

```
virtual HRESULT QueryAboutToBuild(  
    ECodeWarriorCompileChoice choice,  
    long buildID,  
    ICodeWarriorTargetCollection *targetList);
```

choice

The kind of operation the compile performed.

buildID

The ID of the build.

targetList

A pointer to the collection of targets to build.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorCompileChoice” on page 248](#)

[“Using the Collections API” on page 77](#)

---

## QueryDeleteDesign

The CodeWarrior IDE calls this method to inform the plug-in that a design is about to be deleted.

```
virtual HRESULT QueryDeleteDesign(  
    ICodeWarriorDesign *design);
```

design

A pointer to the design to delete.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## QueryUIClose

The CodeWarrior IDE calls this method to inform the plug-in that the IDE is about to close.

```
virtual HRESULT QueryUIClose(  
    ICodeWarriorProject *project);
```

project

A pointer to the project window to close.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

---

**Projects***RevertCompleted*

---

**RevertCompleted**

This method indicates that a reversion (backing up to an earlier state) operation has finished.

```
virtual HRESULT RevertCompleted(void);
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**VisibleChanged**

This method indicates that the project window's visibility has changed (it has been hidden or revealed).

```
virtual HRESULT VisibleChanged(  
    ICodeWarriorProject *project,  
    VARIANT_BOOL fVisible);
```

project

A pointer to the project in question.

fVisible

Set this parameter to true if the project is visible or false if the project is invisible.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

## ICodeWarriorProjectFile

This interface provides the means to manipulate project files..

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">CheckIn</a>	<a href="#">get_Project</a>
<a href="#">CheckOut</a>	<a href="#">get_Targets</a>
<a href="#">get_FileSpec</a>	<a href="#">get_VCSState</a>
<a href="#">get_Name</a>	

---

### CheckIn

This method checks in the project file for the current project.

```
virtual HRESULT CheckIn(void);
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### CheckOut

This method checks out the current project file.

```
virtual HRESULT CheckOut(void);
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_FileSpec

This method gets the file specification for the current project file.

```
virtual HRESULT get_FileSpec(IFileSpec **pval);  
  
pval
```

On return, this parameter contains the address of a pointer to the file specification for the current project file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## get\_Name

This method gets the name of the current project file.

```
virtual HRESULT get_Name(BSTR *pval);  
  
pval
```

On return, this parameter contains the name of the current project file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Project

This method gets the project object for the current project.

```
virtual HRESULT get_Project(  
    ICodeWarriorProject **pval);
```

pval

On return, this parameter contains the address of a pointer to the project object for the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

---

## get\_Targets

This method gets the collection of targets within the current project.

```
virtual HRESULT get_Targets(  
    ICodeWarriorTargetCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to the collection of target files within the current project.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_VCSState

This method gets the current version control system (VCS) status for the current project file.

```
virtual HRESULT get_VCSState(  
    ICodeWarriorVCSState **pval);
```

pval

On return, this parameter contains the address of a pointer to the VCSState object for the current project.

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorVCSState” on page 375](#)

## Project Data Types

### EPluginDataStorageLoc

The EPluginDataStorageLoc enumeration provides constants for where a plug-in’s data is stored.

**Table 15.1 EPluginDataStorageLoc Enumeration**

Constant	Description
kStoreInProjectFile	Store the data in the project file.
kStoreInTargetDataFile	Store the data in a data file associated with the target.
kStoreInProjectSettingsFile	Store the data in the project settings file.

### ECodeWarriorCompileChoice

The ECodeWarriorCompileChoice enumeration provides constants for what kind of operation the compiler performs.



**Table 15.2 ECodeWarriorCompileChoice Enumeration**

Constant	Description
kCWChoiceCheckSyntax	Check the syntax only.
kCWChoicePreprocess	Preprocess only.
kCWChoicePrecompile	Precompile only.
kCWChoiceCompile	Compile normally.
kCWChoiceDisassemble	Disassemble.

### **ECodeWarriorRunMode**

The ECodeWarriorRunMode enumeration provides constants for whether to run the resulting output of a build process and whether to run it in debug mode.

**Table 15.3 ECodeWarriorRunMode Enumeration**

Constant	Description
kCWDontRun	Don't run the application after building.
kCWRun	Run the application after building.
kCWDebug	Run the application in debug mode after building.

### **ECodeWarriorBuildOptions**

The ECodeWarriorBuildOptions enumeration provides constants for whether to build normal or to skip dependencies.

**Table 15.4 ECodeWarriorBuildOptions Enumeration**

Constant	Description
kCWNormalBuild	Build the application normally
kCWSkipDependencies	Skip all dependencies when building.



## **Projects**

*Project Data Types*

---

# Symbols

---

This chapter describes the Symbols API, which you can use to manipulate the various symbols and messages associated with the build process.

This chapter contains the following sections:

- [Symbols API Reference](#)

## Symbols API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorBaseClassInfo](#)
- [ICodeWarriorClass](#)
- [ICodeWarriorDataMember](#)
- [ICodeWarriorMethod](#)
- [ICodeWarriorSourceContext](#)
- [ICodeWarriorSymbol](#)
- [ICodeWarriorSymbolContainer](#)

These interfaces use various data types, as shown in the following section:

- [Symbols Data Types](#)

## ICodeWarriorBaseClassInfo

This interface provides methods to get information about a base class.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_Access</a>	<a href="#">get_IsVirtual</a>
<a href="#">get_BaseClass</a>	

---

### get\_Access

This method gets the access level for the current class.

```
virtual HRESULT get_Access(
    ECodeWarriorAccess *pval);
```

pval

On return, this parameter contains a pointer to the access level.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorAccess” on page 283](#)

---

### get\_BaseClass

This method gets the base class for the current class.

```
virtual HRESULT get_BaseClass(
    ICodeWarriorClass **pval);
```

`pval`

On return, this parameter contains the address of a pointer to the class.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorClass” on page 254](#)

---

## get\_IsVirtual

This method gets whether the base class is virtual.

```
virtual HRESULT get_IsVirtual(  
    BOOL *pval);
```

`pval`

On return, this parameter contains a pointer to a boolean that is set to `true` if the base class is virtual or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorClass

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">FindDataMemberByName</a>	<a href="#">get_IsFinal</a>
<a href="#">FindMethodByName</a>	<a href="#">get_IsPublic</a>
<a href="#">get_BaseClasses</a>	<a href="#">GetMethods</a>
<a href="#">GetDataMembers</a>	<a href="#">GetMethodsWithAccess</a>
<a href="#">GetDataMembersWithAccess</a>	<a href="#">get_SubClasses</a>
<a href="#">get_IsAbstract</a>	

---

## FindDataMemberByName

This method finds a data member within the class, by name.

```
virtual HRESULT FindDataMemberByName(
    BSTR inName,
    ICodeWarriorDataMember **pval);
```

*inName*

The name of the member to find

*pval*

On return, this parameter contains the address of a pointer to the data member specified by the *inName* parameter.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorDataMember” on page 261](#)

---

## FindMethodByName

This method finds a method within the class, by name.

```
virtual HRESULT FindMethodByName(  
    BSTR inName,  
    ICodeWarriorMethod **pval);
```

*inName*

The name of the method to find.

*pval*

On return, this parameter contains the address of a pointer to the data member specified by the *inName* parameter.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorMethod” on page 264](#)

---

## get\_BaseClasses

This method gets a collection of base classes for the current class.

```
virtual HRESULT get_BaseClasses(  
    ICodeWarriorBaseClassCollection **pval);
```

*pval*

On return, this parameter contains the address of a pointer to a collection of base classes.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

**Symbols***GetDataMembers*

---

**GetDataMembers**

This method gets the data members of the class.

```
virtual HRESULT GetDataMembers(  
    BOOL inIncludeInherited,  
    ICodeWarriorDataMemberCollection **pval);
```

*inIncludeInherited*

Set this parameter to `true` to include inherited data members or to `false` to exclude them.

*pval*

On return, this parameter contains the address of a pointer to a collection of the data members.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

**GetDataMembersWithAccess**

This method gets the data members within a class that have a particular kind of access, as determined by the *inAccessMask* parameter.

```
virtual HRESULT GetDataMembersWithAccess(  
    BOOL inIncludeInherited,  
    ECodeWarriorAccess inAccessMask,  
    ICodeWarriorDataMemberCollection **pval);
```

*inIncludeInherited*

Set this parameter to `true` to include inherited data members or to `false` to exclude them.

*inAccessMask*

Set this parameter to one of the constants defined in the



[ECodeWarriorAccess](#) enumeration.

pval

On return, this parameter contains the address of a pointer to a collection of the data members that match the access mask.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_IsAbstract

This method gets whether the class is abstract.

```
virtual HRESULT get_IsAbstract(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that indicates whether the class is abstract.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_IsFinal

This method gets whether the class is final.

```
virtual HRESULT get_IsFinal(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that indicates whether the class is final.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

---

## get\_IsPublic

This method gets whether the class is public.

```
virtual HRESULT get_IsPublic(
    BOOL *pval);
```

*pval*

On return, this parameter contains a pointer to a boolean that indicates whether the class is public.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## GetMethods

This method gets the methods within a class.

```
virtual HRESULT GetMethods(
    BOOL inIncludeInherited,
    ICodeWarriorMethodCollection **pval);
```

*inIncludeInherited*

Set this parameter to `true` to include inherited methods or to `false` to exclude them.

*pval*

On return, this parameter contains the address of a pointer to a collection of the methods.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## GetMethodsWithAccess

This method gets the data members within a class that have a particular kind of access, as determined by the `inAccessMask` parameter.

```
virtual HRESULT GetMethodsWithAccess(  
    BOOL inIncludeInherited,  
    ECodeWarriorAccess inAccessMask,  
    ICodeWarriorMethodCollection **pval);
```

`inIncludeInherited`

Set this parameter to `true` to include inherited members or to `false` to exclude them.

`inAccessMask`

Set this parameter to one of the constants defined in the [ECodeWarriorAccess](#) enumeration.

`pval`

On return, this parameter contains the address of a pointer to a collection of the methods that match the access mask.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## Symbols

*get\_SubClasses*

---

### get\_SubClasses

This method gets a collection containing the subclasses of the class.

```
virtual HRESULT get_SubClasses(  
    ICodeWarriorClassCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to a collection of the current class's subclasses.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## ICodeWarriorDataMember

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_Access</a>	<a href="#">get_IsTransient</a>
<a href="#">get_IsFinal</a>	<a href="#">get_IsVolatile</a>
<a href="#">get_IsStatic</a>	

---

### get\_Access

This method gets the access type of the current data member.

```
virtual HRESULT get_Access(
    ECodeWarriorAccess *pval);
```

pval

On return, this parameter contains a pointer to the access type.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorAccess” on page 283](#)

**Symbols***get\_IsFinal*

---

**get\_IsFinal**

This method gets whether the current data member is final.

```
virtual HRESULT get_IsFinal(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the data member is final or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**get\_IsStatic**

This method gets whether the current data member is static.

```
virtual HRESULT get_IsStatic(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the data member is static or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

---

## get\_IsTransient

This method gets whether the current data member is transient.

```
virtual HRESULT get_IsTransient(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the data member is transient or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_IsVolatile

This method gets whether the current data member is volatile.

```
virtual HRESULT get_IsVolatile(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the data member is volatile or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorMethod

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_Access</a>	<a href="#">get_IsInline</a>
<a href="#">get_IsAbstract</a>	<a href="#">get_IsNative</a>
<a href="#">get_IsConst</a>	<a href="#">get_IsStatic</a>
<a href="#">get_IsConstructor</a>	<a href="#">get_IsSynchronized</a>
<a href="#">get_IsDestructor</a>	<a href="#">get_IsVirtual</a>

---

## get\_Access

This method gets the access restriction for the current method.

```
virtual HRESULT get_Access(
    ECodeWarriorAccess *pval);
```

pval

On return, this pointer contains a pointer to the access restriction for the current method.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorAccess” on page 283](#)



---

## get\_IsAbstract

This method gets whether the current method is abstract.

```
virtual HRESULT get_IsAbstract(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is abstract or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## get\_IsConst

This method gets whether the current method has been declared as constant.

```
virtual HRESULT get_IsConst(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is abstract or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Symbols***get\_IsConstructor*

---

**get\_IsConstructor**

This method gets whether the current method is one of a class's constructors.

```
virtual HRESULT get_IsConstructor(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is a constructor or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**get\_IsDestructor**

This method gets whether the current method is one of a class's destructors.

```
virtual HRESULT get_IsDestructor(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is a destructor or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

---

## get\_IsInline

This method gets whether the current method is inline.

```
virtual HRESULT get_IsInline(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is inline or `false` if not.

Returns Nothing

---

## get\_IsNative

This method gets whether the current method is native.

```
virtual HRESULT get_IsNative(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is native or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## Symbols

*get\_IsStatic*

---

### get\_IsStatic

This method gets whether the current method is static.

```
virtual HRESULT get_IsStatic(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is static or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

### get\_IsSynchronized

This method gets whether the current method is synchronized.

```
virtual HRESULT get_IsSynchronized(  
    BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is synchronized or `false` if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## get\_IsVirtual

This method gets whether the current method is virtual.

```
virtual HRESULT get_IsVirtual(  
    BOOL *pval);
```

`pval`

On return, this parameter contains a pointer to a boolean that is set to `true` if the method is virtual or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorSourceContext

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_EndOffset</a>	<a href="#">put_EndOffset</a>
<a href="#">get_FileSpec</a>	<a href="#">put_FileSpec</a>
<a href="#">get_IsDefined</a>	<a href="#">put_StartOffset</a>
<a href="#">get_StartOffset</a>	

---

### get\_EndOffset

This method gets the number of characters from the top of a source file to the end of a specified symbol (usually one identified by the compiler as problematic).

```
virtual HRESULT get_EndOffset(
    long *pval)
```

pval

On return, this parameter contains a pointer to a long integer that indicates how many characters from the top of the file to the end of the symbol in question.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## get\_FileSpec

This method gets the file specification for a specified symbol (usually one identified by the compiler as problematic).

```
virtual HRESULT get_FileSpec(  
    IFileSpec **pval)
```

*pval*

On return, this parameter contains the address of a pointer to the file specification for the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## get\_IsDefined

This method indicates whether the symbol in question has been defined.

```
virtual HRESULT get_IsDefined(  
    BOOL *pval)
```

*pval*

On return, this parameter contains a pointer to a boolean that is set to `true` if the symbol in question has been defined or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## Symbols

*get\_StartOffset*

---

### get\_StartOffset

This method gets the number of characters from the top of a source file to the start of a specified symbol (usually one identified by the compiler as problematic).

```
virtual HRESULT get_StartOffset(  
    long *pval)
```

pval

On return, this parameter contains a pointer to a long integer that indicates how many characters from the top of the file to the start of the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### put\_EndOffset

This method sets the number of characters to the end of a symbol in a source file.

```
virtual HRESULT put_EndOffset(  
    long pval)
```

pval

The number of characters from the top of the file to the end of the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



---

## put\_FileSpec

This method sets the file specification for a symbol.

```
virtual HRESULT put_FileSpec(  
    IFileSpec *pval)
```

pval

A pointer to a file specification.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## put\_StartOffset

This method sets the number of characters to the start of a symbol in a source file.

```
virtual HRESULT put_StartOffset(  
    long pval)
```

pval

The number of characters from the top of the file to the start of the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorSymbol

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_Class</a>	<a href="#">get_DefinitionLocation</a>
<a href="#">get_Container</a>	<a href="#">get_Name</a>
<a href="#">get_DeclarationLocation</a>	<a href="#">get_SimpleName</a>

---

### get\_Class

This method gets the class in which a symbol appears

```
virtual HRESULT get_Class(
    ICodeWarriorClass **pval);
```

pval

On return, this parameter contains the address of a pointer to the class containing the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorClass” on page 254](#)

## get\_Container

This method gets the container for a symbol.

```
virtual HRESULT get_Container(
    ICodeWarriorSymbolContainer **pval);
```

pval

On return, this parameter contains the address of a pointer to the container that holds the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSymbolContainer” on page 278](#)

---

## get\_DeclarationLocation

This method gets the location where the symbol was declared.

```
virtual HRESULT get_DeclarationLocation(
    ICodeWarriorSourceContext **pval);
```

pval

On return, this parameter contains the address of a pointer to the source context where the symbol in question was declared.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSourceContext” on page 270](#)

**Symbols***get\_DefinitionLocation*

---

---

**get\_DefinitionLocation**

This method gets the location where the symbol was defined.

```
virtual HRESULT get_DefinitionLocation(  
    ICodeWarriorSourceContext **pval);
```

pval

On return, this parameter contains the address of a pointer to the source context where the symbol in question was defined.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSourceContext” on page 270](#)

---

**get\_Name**

This method gets the fully qualified name of the symbol.

```
virtual HRESULT get_Name(  
    BSTR *pval);
```

pval

On return, this parameter contains the name of the symbol in question.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**get\_SimpleName**

This method gets the simple name of the symbol.

```
virtual HRESULT get_SimpleName(  
    BSTR *pval);
```

---

`pval`

On return, this parameter contains the simple name of the symbol in question.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorSymbolContainer

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">AddComponentAttachment</a>	<a href="#">get_Target</a>
<a href="#">FindClass</a>	<a href="#">RemoveComponentAttachment</a>
<a href="#">FindClassInFile</a>	<a href="#">ShowSymbolDeclaration</a>
<a href="#">get_ClassList</a>	<a href="#">ShowSymbolDefinition</a>

---

### AddComponentAttachment

This method lets you add a component attachment to a symbol container.

```
virtual HRESULT AddComponentAttachment(  
    CLSID *attachmentCLSID);
```

attachmentCLSID

A pointer to the component attachment to add.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### FindClass

This method finds a class, given its name.

```
virtual HRESULT FindClass(  
    BSTR inClassName,  
    ICodeWarriorClass **outClass);
```

inClassName

The name of the class to find.

outClass

On return, this parameter contains the address of a pointer to the class.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorClass” on page 254](#)

---

## FindClassInFile

This method finds a file within a class, given the class’s name and a file specification.

```
virtual HRESULT FindClassInFile(  
    BSTR inClassName,  
    IFileSpec *inSpec,  
    ICodeWarriorClass **outClass);
```

inClassName

The name of the class.

inSpec

A pointer to a file specification.

outClass

On return, this parameter contains the address of a pointer to the class.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

[“ICodeWarriorClass” on page 254](#)

---

**Symbols***get\_ClassList*

---

---

**get\_ClassList**

This method gets a list of the classes referenced by the symbol container.

```
virtual HRESULT get_ClassList(  
    ICodeWarriorClassCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to the class list.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

**get\_Target**

This method gets the target associated with the symbol container (that is, the target for which the build process generated the symbols in the symbol container).

```
virtual HRESULT get_Target(  
    ICodeWarriorTarget **pval);
```

pval

On return, this parameter contains the address of a pointer to the target associated with the current symbol container.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTarget” on page 286](#)



---

## RemoveComponentAttachment

This method removes a component attachment from the symbol container.

```
virtual HRESULT RemoveComponentAttachment(  
    CLSID *attachmentCLSID);
```

attachmentCLSID

A pointer to the component attachment to remove.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## ShowSymbolDeclaration

This method shows the user where a specified symbol is located. It may also let the user edit the symbol in that location.

```
virtual HRESULT ShowSymbolDeclaration(  
    ICodeWarriorSymbol *inSymbol,  
    BOOL inForEditing,  
    ECodeWarriorShowSymbolLocation inLocation);
```

inSymbol

The symbol to show the user.

inForEditing

Set this parameter to `true` to let the user edit the line on which the symbol appears or `false` to prevent editing.

inLocation

Where to show the symbol.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSymbol” on page 274](#)

**Symbols***ShowSymbolDefinition*

---

[“ECodeWarriorShowSymbolLocation” on page 283](#)

---

## ShowSymbolDefinition

This method shows where the symbol in question is defined.

```
virtual HRESULT ShowSymbolDefinition(  
    ICodeWarriorSymbol *inSymbol,  
    BOOL inForEditing,  
    ECodeWarriorShowSymbolLocation inLocation);
```

*inSymbol*

The symbol to show the user.

*inForEditing*

Set this parameter to `true` to let the user edit the line on which the symbol appears or `false` to prevent editing.

*inLocation*

Where to show the symbol.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSymbol” on page 274](#)

[“ECodeWarriorShowSymbolLocation” on page 283](#)

## Symbols Data Types

### ECodeWarriorAccess

The ECodeWarriorAccess enumeration provides constants for member access levels.

Constant	Definition
kAccessNone	No Access.
kPublicAccess	Access to public members.
kProtectedAccess	Access to protected (and public) members.
kPrivateAccess	Access to private (and protected and public members).
kAccessAll	Access to all members.

### ECodeWarriorShowSymbolLocation

The ECodeWarriorAccess enumeration provides constants for where a symbol can be shown.

Constant	Definition
kShowInEditor	The symbol can be shown in the editor.
kShowInBrowser	The symbol can be shown in the symbol browser.
kUsePreferenceToShow	The symbol conforms to user settings for where to show symbols.



## **Symbols**

*Symbols Data Types*

---

# Targets

---

This chapter shows how to use the Targets API to manage operations with targets in the CodeWarrior IDE.

This chapter contains the following sections:

- [Targets API Overview](#)
- [Targets API Reference](#)

## Targets API Overview

The Targets API is a set of interfaces that allows a plug-in to create and manipulate targets in the CodeWarrior IDE.

## Targets API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorTarget](#)
- [ICodeWarriorTargetFile](#)
- [ICodeWarriorTargetOutput](#)
- [ICodeWarriorSubTarget](#)
- [ICodeWarriorSubProjectTarget](#)

These interfaces use constants from enumerations described in the following section:

- [Targets Data Types](#)

## ICodeWarriorTarget

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">AddFile</a>	<a href="#">get_Design</a>
<a href="#">AddFile2</a>	<a href="#">GetLinkerName</a>
<a href="#">AddFile2ByFileSpec</a>	<a href="#">get_Name</a>
<a href="#">AddFile2ByFileSpecCollection</a>	<a href="#">GetNamedPluginData</a>
<a href="#">AddFileByFileSpec</a>	<a href="#">get_Project</a>
<a href="#">AddFileByFileSpecCollection</a>	<a href="#">get_ProjectFileCollection</a>
<a href="#">AddSubTarget</a>	<a href="#">GetProjectFileFromFileSpec</a>
<a href="#">AddUserTree</a>	<a href="#">GetSubProjects</a>
<a href="#">Build</a>	<a href="#">get_SubTargets</a>
<a href="#">BuildAgainstSubProjectTarget</a>	<a href="#">get_TargetFileCollection</a>
<a href="#">BuildAndWaitToComplete</a>	<a href="#">GetTargetFileForProjectFile</a>
<a href="#">BuildAndWaitToCompleteWithOptions</a>	<a href="#">GetTargetOutput</a>
<a href="#">BuildWithOptions</a>	<a href="#">get_UserTrees</a>
<a href="#">CompileFiles</a>	<a href="#">LinkAgainstSubProjectTarget</a>
<a href="#">CompileFilesAndWaitToComplete</a>	<a href="#">LinkAgainstSubTarget</a>
<a href="#">CompileFilesWithChoice</a>	<a href="#">put_BrowserEnabled</a>
<a href="#">CreateUserTree</a>	<a href="#">put_Name</a>
<a href="#">FindAndAddFile</a>	<a href="#">RemoveNamedPluginData</a>
<a href="#">FindAndAddFile2</a>	<a href="#">RemoveObjectCode</a>
<a href="#">FindAndAddFile2ByCollection</a>	<a href="#">RemoveObjectCodeWithOptions</a>
<a href="#">FindAndAddFileByCollection</a>	<a href="#">RemoveUserTree</a>
<a href="#">get_AccessPaths</a>	<a href="#">SetNamedPluginData</a>

---

<a href="#">get_BrowserDB</a>	<a href="#">SetupDebugging</a>
<a href="#">get_BrowserEnabled</a>	<a href="#">SynchronizeStatus</a>

---

---

## AddFile

This method adds a file to the current target.

```
virtual HRESULT AddFile(  
    BSTR path,  
    BSTR groupPath,  
    ICodeWarriorProjectFile **projectFile);
```

path

The path to the file to add.

groupPath

The path to the group to which to add the file.

projectFile

On return, this parameter contains the address of a pointer to the project file associated with the added file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

---

## AddFile2

This method adds a file to the current target and set link flags on the file.

```
virtual HRESULT AddFile2(  
    BSTR path,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,
```

## Targets

### AddFile2ByFileSpec

---

```
ICodeWarriorProjectFile **projectFile) = 0;
```

path

The path to the file to add.

groupPath

The path to the group to which to add the file.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

projectFile

On return, this parameter contains the address of a pointer to the project file associated with the added file.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

[“ECodeWarriorLinkFlags” on page 145](#)

---

## AddFile2ByFileSpec

This method adds a file to the current target, by using a file specification object, and set link flags on the file.

```
virtual HRESULT AddFile2ByFileSpec(
    IFileSpec __RPC_FAR *fileSpec,
    BSTR groupPath,
    ECodeWarriorLinkFlags linkFlags,
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

The path to the file to add.

groupPath

The path to the group to which to add the file.

---



linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

projectFile

On return, this parameter contains the address of a pointer to the project file associated with the added file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)  
[“ICodeWarriorProjectFile” on page 245](#)  
[“ECodeWarriorLinkFlags” on page 145](#)

---

## AddFile2ByFileSpecCollection

This method adds a collection of files to the current target and set link flags on the files.

```
virtual HRESULT AddFile2ByFileSpecCollection(  
    IFileSpecCollection __RPC_FAR *inCollection,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,  
    int *pFilesAdded) = 0;
```

path

The path to the file to add.

groupPath

The path to the group to which to add the file.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

## Targets

### AddFileByFileSpec

---

projectFile

On return, this parameter contains the address of a pointer to the project file associated with the added file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

[“ICodeWarriorProjectFile” on page 245](#)

[“ECodeWarriorLinkFlags” on page 145](#)

---

## AddFileByFileSpec

This method adds a file to the current target, by file specification.

```
virtual HRESULT AddFileByFileSpec(
    IFileSpec *fileSpec,
    BSTR groupPath,
    ICodeWarriorProjectFile **projectFile);
```

path

The path to the file to add.

groupPath

The path to the group to which to add the file.

projectFile

On return, this parameter contains the address of a pointer to the project file associated with the added file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

[“ICodeWarriorProjectFile” on page 245](#)

## AddFileByFileSpecCollection

This method adds all the files in a collection of file specifications to the current target.

```
virtual HRESULT AddFileByFileSpecCollection(  
    IFileSpecCollection *inCollection,  
    BSTR groupPath,  
    int *pFilesAdded);
```

`inCollection`

A pointer to the collection of file specifications that defines the files to add.

`groupPath`

The path to the group to which to add the files.

`pFilesAdded`

On return, this parameter contains a pointer to an integer holding the number of files added to the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## AddSubTarget

CodeWarrior targets can contain other CodeWarrior targets. This method adds a target within the current target.

```
virtual HRESULT AddSubTarget(  
    ICodeWarriorTarget *target,  
    VARIANT_BOOL linkAgainstOutput);
```

`target`

A pointer to the target to be added.

## Targets

### AddUserTree

---

linkAgainstOutput

true to link against the output of the added subtarget or false if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTarget” on page 286](#)

---

### AddUserTree

This method adds an existing user tree to the current target.

```
virtual HRESULT AddUserTree(
    ICodeWarriorUserTree *pval) = 0;
```

pval

A pointer to the user tree to add to the application.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorUserTree” on page 26](#)

---

### Build

This method tells the CodeWarrior IDE to build the current target.

```
virtual HRESULT Build(
    long *cookie);
```

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## BuildAgainstSubProjectTarget

This method controls whether to build a target within a subproject.

```
virtual HRESULT BuildAgainstSubProjectTarget(  
    ICodeWarriorSubProjectTarget *target,  
    VARIANT_BOOL val) = 0;
```

target

A pointer to the target to build.

val

true to build the target or false to not build it.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSubProjectTarget” on page 329](#)

---

## BuildAndWaitToComplete

This method starts a build of the current project and has the IDE wait to gather all messages from the build process.

```
virtual HRESULT BuildAndWaitToComplete(  
    ICodeWarriorBuildMessages **buildMessages);
```

buildMessages

On return, this parameter contains the address of a pointer to the messages created by the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorBuildMessages” on page 202](#)

## Targets

### *BuildAndWaitToCompleteWithOptions*

---

## BuildAndWaitToCompleteWithOptions

This method builds the current project with one of the options specified in the [ECodeWarriorBuildOptions](#) enumeration. This method accumulates all the messages from the build process before returning.

```
virtual HRESULT BuildAndWaitToCompleteWithOptions(
    ECodeWarriorBuildOptions options,
    ICodeWarriorBuildMessages **buildMessages) = 0;
```

options

The build options to use with this build.

buildMessages

On return, this parameter contains the address of a pointer to the build messages created by the build process.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorBuildOptions” on page 249](#)

---

## BuildWithOptions

This method builds the current target with one of the options specified in the [ECodeWarriorBuildOptions](#) enumeration.

```
virtual HRESULT BuildWithOptions(
    ECodeWarriorBuildOptions options,
    ECodeWarriorRunMode runMode,
    long *cookie) = 0;
```

options

The build options to use with this build.

runmode

Whether to run the resulting program after building it and, if so, whether to run it in debug mode. The [ECodeWarriorRunMode](#) enumeration contains the constants that define this parameter.

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorBuildOptions” on page 249](#)  
[“ECodeWarriorRunMode” on page 249](#)

---

## CompileFiles

Use this method to compile the current target.

```
virtual HRESULT CompileFiles(  
    ICodeWarriorProjectFileCollection *collection,  
    long *cookie);
```

collection

A pointer of type ICodeWarriorProjectFileCollection indicating the collection of files to compile.

cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

**Targets***CompileFilesAndWaitToComplete*

---

---

## CompileFilesAndWaitToComplete

Use this method to compile the current target and return all the build messages created by the compiler.

```
virtual HRESULT CompileFilesAndWaitToComplete(  
    ICodeWarriorProjectFileCollection *collection,  
    ICodeWarriorBuildMessages **buildMessages);
```

collection

A pointer of type `ICodeWarriorProjectFileCollection` indicating the collection of files to compile.

buildMessages

On return, this parameter contains the address of a pointer to the build messages generated by the compiler.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## CompileFilesWithChoice

This method compiles a collection of project files with one of the options specified in the [ECodeWarriorCompileChoice](#) enumeration.

```
virtual HRESULT CompileFilesWithChoice(  
    ICodeWarriorProjectFileCollection *collection,  
    ECodeWarriorCompileChoice compileChoice,  
    long *cookie) = 0;
```

collection

A pointer to a collection of file projects to compile.

compileChoice

The kind of compilation the compiler should perform.



cookie

On return, this parameter contains a unique identifier for the build process.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorCompileChoice” on page 248](#)

---

## CreateUserTree

This method creates a new user tree.

```
virtual HRESULT CreateUserTree(  
    BSTR displayName,  
    BSTR value,  
    EUserDefinedTree type,  
    BSTR keyName,  
    ICodeWarriorUserTree **pVal) = 0;
```

displayName

The name of the user tree that will appear in the IDE.

value

The value string of the user tree.

type

The type of the tree, which must be one of the values specified by the [EUserDefinedTree](#) Tree enumeration.

keyName

The key name of the user tree.

pval

On return, this parameter contains the address of a pointer to the new user tree.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**Targets***FindAndAddFile*

---

See Also [“EUserDefinedTree” on page 30](#)  
[“ICodeWarriorUserTree” on page 26](#)

---

**FindAndAddFile**

This method finds a file and adds it to the current target.

```
virtual HRESULT FindAndAddFile(  
    BSTR path,  
    BSTR groupPath,  
    ICodeWarriorProjectFile **projectFile);
```

path

Either the absolute (fully qualified) path to the file you want to add to the design or just the name of the file. If you provide just the file name, the IDE searches the access paths and adds the first file of that name that it finds. If you want to add two one files with identical names, use the fully qualified path to each one.

groupPath

The absolute path to the group within which the new file should be added.

projectFile

On return, this parameter contains the address of a pointer to the project file that contains the current target..

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

## FindAndAddFile2

This method finds a file and adds it to the design. This method also lets you set link flags on the file.

```
virtual HRESULT FindAndAddFile2(  
    BSTR path,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,  
    ICodeWarriorProjectFile **projectFile) = 0;
```

path

Either the absolute (fully qualified) path to the file you want to add to the design or just the name of the file. If you provide just the file name, the IDE searches the access paths and adds the first file of that name that it finds. If you want to add two one files with identical names, use the fully qualified path to each one.

groupPath

The absolutepath to the group within which the new file should be added.

linkFlags

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

projectFile

On return, this parameter contains the address of a pointer to the project file to which the file was added.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

**Targets***FindAndAddFile2ByCollection*

---

---

## FindAndAddFile2ByCollection

This method adds a named collection of files to the current target, setting link flags on all the added files in the process.

```
virtual HRESULT FindAndAddFile2ByCollection(  
    IBSTRCollection *inCollection,  
    BSTR groupPath,  
    ECodeWarriorLinkFlags linkFlags,  
    int *pFilesAdded) = 0;
```

`inCollection`

The name of the collection of files to add to the current target.

`groupPath`

The absolute path to the group within which the new file should be added.

`linkFlags`

A value in the range defined by the [ECodeWarriorLinkFlags](#) enumeration, representing how the linker should link this file.

`pFilesAdded`

On return, this parameter contains a pointer to the number of files added.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

---

## FindAndAddFileByCollection

This method adds a collection of files to the current target.

```
virtual HRESULT FindAndAddFileByCollection(  
    IBSTRCollection *inCollection,  
    BSTR groupPath,  
    int *pFilesAdded);
```

*inCollection*

The file collection to add.

*groupPath*

The path to the group within which the new files should be added.

*pFilesAdded*

On return, this parameter contains a pointer to an integer holding the number of files added to the target.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_AccessPaths

This method gets the access paths for the current target.

```
virtual HRESULT get_AccessPaths(  
    ICodeWarriorAccessPaths **pval);
```

*pval*

On return, this parameter contains the address of a pointer to the access paths of the current target.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Targets

*get\_BrowserDB*

---

See Also [“ICodeWarriorAccessPaths” on page 19](#)

---

## get\_BrowserDB

This method gets the symbols created during the most recent build of the current target.

```
virtual HRESULT get_BrowserDB(
    ICodeWarriorSymbolContainer **catalog);
```

catalog

On return, this parameter contains the address of a parameter to the list of symbols generated by the most recent build of the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSymbolContainer” on page 278](#)

---

## get\_BrowserEnabled

This method gets whether the symbol browser is enabled.

```
virtual HRESULT get_BrowserEnabled(
    VARIANT_BOOL *fEnabled);
```

fEnabled

On return, this parameter contains a pointer to a boolean that is set to `true` if the symbol browse is enabled or `false` otherwise.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

---

## get\_Design

This method gets the design associated with the current target.

```
virtual HRESULT get_Design(  
    ICodeWarriorDesign **design);
```

design

On return, this parameter contains the address of a pointer to the design associated with the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorDesign” on page 130](#)

---

## GetLinkerName

This method gets the name of the current linker.

```
virtual HRESULT GetLinkerName(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of the current linker.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Name

This method gets the name of the current target.

```
virtual HRESULT get_Name(  
    BSTR *pval);
```

---

## Targets

### *GetNamedPluginData*

---

pval

On return, this parameter contains the name of the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetNamedPluginData

This method gets the data from a plug-in, specified by name.

```
virtual HRESULT GetNamedPluginData(
    BSTR resourceName,
    EPluginDataStorageLoc storeIn,
    IStream **pluginData);
```

resourceName

The name of the plug-in from which to get data.

storeIn

The location in which the data is stored.

pluginData

On return, this parameter contains the address of a pointer to the data from the specified plug-in.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## get\_Project

This method gets the project associated with the current target.

```
virtual HRESULT get_Project(
    ICodeWarriorProject **project);
```



project

On return, this parameter contains the address of a pointer to the project associated with the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProject” on page 216](#)

---

## get\_ProjectFileCollection

This method gets the project file collection to which the project file associated with the current target belongs.

```
virtual HRESULT get_ProjectFileCollection(  
    ICodeWarriorProjectFileCollection  
    **projectFileCollection);
```

projectFileCollection

On return, this parameter contains the address of a pointer to the project file collection associated with the project file to which the current target belongs.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## GetProjectFileFromFileSpec

This method gets a project file, by file specification.

```
virtual HRESULT GetProjectFileFromFileSpec(  
    IFileSpec *fileSpec,  
    ICodeWarriorProjectFile **projectFile);
```

## Targets

### *GetSubProjects*

---

`fileSpec`

A pointer to the file specification.

`projectFile`

On return, this parameter contains the address of a pointer to the project file specified in the `fileSpec` parameter.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## GetSubProjects

This method gets a collection of the projects within the current target.

```
virtual HRESULT GetSubProjects(
    ICodeWarriorSubProjectCollection
    **subProjectList) = 0;
```

`subProjectList`

On return, this parameter contains the address of a pointer to the a collection of subprojects.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_SubTargets

A CodeWarrior target can contain other CodeWarrior targets. This method gets the targets contained within the current target.

```
virtual HRESULT get_SubTargets(
    ICodeWarriorSubTargetCollection
```

---

```
**subTargetList);
```

subTargetList

On return, this parameter contains the address of a pointer to the collection of targets within the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_TargetFileCollection

This method gets the collection of target files that the current target contains.

```
virtual HRESULT get_TargetFileCollection(  
    ICodeWarriorTargetFileCollection  
    **targetFileCollection);
```

targetFileCollection

On return, this parameter contains the address of a pointer to the collection of target files within the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## GetTargetFileForProjectFile

This method gets the target file for a specified project file.

```
virtual HRESULT GetTargetFileForProjectFile(  
    ICodeWarriorProjectFile *projectFile,  
    ICodeWarriorTargetFile **targetFile);
```

---

## Targets

### *GetTargetOutput*

---

projectFile

A pointer to the project file for which to get the target file.

targetFile

On return, this parameter contains the address of a pointer to the target file associated with the specified project file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorProjectFile” on page 245](#)

[“ICodeWarriorTargetFile” on page 317](#)

---

## GetTargetOutput

This method gets the output of the current target.

```
virtual HRESULT GetTargetOutput(
    ICodeWarriorTargetOutput **targetOutput);
```

targetOutput

On return, this parameter contains the address of a pointer to the output of the current target.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTargetOutput” on page 325](#)

---

## get\_UserTrees

This method gets the user trees, as a collection of trees.

```
virtual HRESULT get_UserTrees(
    ICodeWarriorUserTreeCollection **pval);
```

---

pval

On return, this parameter contains the address of a pointer to a collection of trees that constitute the user trees.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## Targets

### *LinkAgainstSubProjectTarget*

---

## LinkAgainstSubProjectTarget

This method specifies whether to link against the target produced by a particular subproject within the current target.

```
virtual HRESULT LinkAgainstSubProjectTarget(
    ICodeWarriorSubProjectTarget *target,
    VARIANT_BOOL val) = 0;
```

target

The target (produced by a subproject) to link against (or not).

val

true to link against the target specified in target or false if not.

See Also [“ICodeWarriorSubProjectTarget” on page 329](#)

---

## LinkAgainstSubTarget

This method specifies whether to link against a particular subtarget within the current target.

```
virtual HRESULT LinkAgainstSubTarget(
    ICodeWarriorSubTarget *target,
    VARIANT_BOOL val) = 0;
```

target

The subtarget to link against (or not).

val

true to link against the target specified in target or false if not.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorSubTarget” on page 327](#)

---

## put\_BrowserEnabled

This method sets whether the symbol browser is enabled.

```
virtual HRESULT put_BrowserEnabled(  
    VARIANT_BOOL fEnabled);
```

fEnabled

Set this parameter to `true` to enable the symbol browser or `false` to disable it.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## put\_Name

This target assigns the name of the current target.

```
virtual HRESULT put_Name(  
    BSTR pval);
```

pval

The new name for the current target.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## RemoveNamedPluginData

This method removes the data from a plug-in specified by name.

```
virtual HRESULT RemoveNamedPluginData(  
    BSTR resourceName,  
    EPluginDataStorageLoc storeIn);
```

## Targets

### *RemoveObjectCode*

---

resourceName

The name of the plug-in from which to remove data.

storeIn

The location where the data is stored.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## RemoveObjectCode

This method removes the object code created by building the current target.

```
virtual HRESULT RemoveObjectCode(
    VARIANT_BOOL deleteDataFiles);
```

deleteDataFiles

Set this parameter to true to delete the data files associated with the object data or false to retain them.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## RemoveObjectCodeWithOptions

This method removes the object code from the current target. It also lets you choose whether to remove the object code from all the targets created by subprojects within the current target and whether to delete any associated data files.

```
virtual HRESULT RemoveObjectCodeWithOptions(
    ECodeWarriorWhichTargetOptions
    whichTargetOfSubprojects,
    VARIANT_BOOL recurseSubProject,
```

---



```
VARIANT_BOOL deleteDataFiles) = 0;
```

## Targets

### RemoveUserTree

---

`whichTargetOfSubprojects`

A value with the range defined by the [ECodeWarriorWhichTargetOptions](#) enumeration that specifies whether to remove the object code from all targets or only the current target.

`recurseSubProject`

Set this parameter to `true` to remove the object code within all the subprojects that match the `whichTargetOfSubprojects` parameter or `false` to leave the object code in the subprojects.

`deleteDataFiles`

Set this parameter to `true` to delete data files associated with the current target or `false` to retain the data files.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorWhichTargetOptions” on page 331](#)

---

## RemoveUserTree

This method removes a specified user tree.

```
virtual HRESULT RemoveUserTree(
    ICodeWarriorUserTree *pval) = 0;
```

`pval`

A pointer to the user tree to remove.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorUserTree” on page 26](#)

## SetNamedPluginData

This method assigns data to a plug-in specified by name.

```
virtual HRESULT SetNamedPluginData(  
    BSTR resourceName,  
    EPluginDataStorageLoc storeIn,  
    IStream *pluginData);
```

resourceName

The name of the plug-in to which to assign data.

storeIn

The location in which to store the data.

pluginData

A pointer to the data to store.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“EPluginDataStorageLoc” on page 248](#)

---

## SetupDebugging

This method sets whether debugging is enabled.

```
virtual HRESULT SetupDebugging(  
    VARIANT_BOOL inTurnOn);
```

inTurnOn

Set this parameter to `true` to enable debugging or `false` to disable debugging.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## Targets

### *SynchronizeStatus*

---

## SynchronizeStatus

---

This method synchronizes the status of the target, so that all parts of the target have the same date.

```
virtual HRESULT SynchronizeStatus(void);
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorTargetFile

This interface lets a plug-in work target files.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_DebugInfo</a>	<a href="#">get_Target</a>
<a href="#">get_Dependencies</a>	<a href="#">get_WeakImport</a>
<a href="#">get_Dependents</a>	<a href="#">put_DebugInfo</a>
<a href="#">get_FileSpec</a>	<a href="#">put_InitBefore</a>
<a href="#">get_InitBefore</a>	<a href="#">put_MergeLibrary</a>
<a href="#">get_MergeLibrary</a>	<a href="#">put_WeakImport</a>
<a href="#">get_Name</a>	

---

### get\_DebugInfo

This method gets whether a debugger can get debugging information from the current target file.

```
virtual HRESULT get_DebugInfo(
    VARIANT_BOOL *value);
```

value

On return, this parameter contains a boolean that is set to `true` if a debugging application can get debugging information from the current target file and `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Targets

### *get\_Dependencies*

---

## get\_Dependencies

This method returns a collection containing the dependencies (the files on which this target depends) for the current target file.

```
virtual HRESULT get_Dependencies(
    ICodeWarriorTargetFileCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to a collection of files that form the dependencies for the current target file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## get\_Dependents

This method returns a collection containing the dependents (the files that depend on this target) for the current target file.

```
virtual HRESULT get_Dependents(
    ICodeWarriorTargetFileCollection **pval);
```

pval

On return, this parameter contains the address of a pointer to a collection of files that form the dependents for the current target file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

---

## get\_FileSpec

This method gets the file specification for the current target file.

```
virtual HRESULT get_FileSpec(  
    IFileSpec **pval);
```

pval

On return, this parameter contains the address of a pointer to the file specification for the current target file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## get\_InitBefore

This method gets the state of the CWInitBefore flag.

```
virtual HRESULT get_InitBefore(  
    VARIANT_BOOL *value);
```

value

On return, this parameter contains a pointer to a boolean set to true if the CWInitBefore flag is set to true or false if the CWInitBefore flag is set to false.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

## Targets

### *get\_MergeLibrary*

---

#### get\_MergeLibrary

This method gets whether the build process should merge with libraries when building this target file.

```
virtual HRESULT get_MergeLibrary(
    VARIANT_BOOL *value);
```

value

On return, this parameter contains a boolean that is set to `true` if the current target file should be merged with libraries during the next build or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

#### get\_Name

This method gets the name of the current target file.

```
virtual HRESULT get_Name(
    BSTR *pval);
```

pval

On return, this parameter contains the name of the current target file.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

#### get\_Target

This method gets the target object associated with the current target file.

```
virtual HRESULT get_Target(
```

---



```
ICodeWarriorTarget **pval);
```

pval

On return, this parameter contains the address of a pointer to the target object associated with the current target file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorTarget” on page 286](#)

---

## get\_WeakImport

This method gets whether this target file should be built with the Weak Import option (available only on the Mac OS).

```
virtual HRESULT get_WeakImport(  
    VARIANT_BOOL *value);
```

value

On return, this parameter contains a boolean that is set to `true` if the current target file should be build with the Weak Import option and `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_DebugInfo

This method sets whether a debugger can get debugging information from the current target file.

```
virtual HRESULT put_DebugInfo(  
    VARIANT_BOOL value);
```

value

Set this parameter to `true` if a debugging application can get

---



## Targets

*put\_DebugInfo*

---

debugging information from the current target file and `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## put\_InitBefore

This method sets the `CWInitBefore` flag.

```
virtual HRESULT put_InitBefore(  
    VARIANT_BOOL value);
```

value

`true` to set the `CWInitBefore` flag to true or `false` to set `CWInitBefore` the flag to false.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorLinkFlags” on page 145](#)

---

## put\_MergeLibrary

This method sets whether the build process should merge with libraries when building this target file.

```
virtual HRESULT put_MergeLibrary(  
    VARIANT_BOOL value);
```

value

Set this parameter to `true` if the current target file should be merged with libraries during the next build or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Targets

*put\_WeakImport*

---

### put\_WeakImport

---

This method sets whether this target file should be built with the Weak Import option (available only on the Mac OS).

```
virtual HRESULT put_WeakImport(  
    VARIANT_BOOL value);
```

value

Set this parameter to `true` if the current target file should be build with the Weak Import option and `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorTargetOutput

.This interface gets information about the output resulting from building the current target.

The following methods are available for your use:

<a href="#">get_FileSpec</a>	<a href="#">get_OutputKind</a>
------------------------------	--------------------------------

---

---

### get\_FileSpec

This method gets the file specification for the output file that is created when the current target is built.

```
virtual HRESULT get_FileSpec(  
    IFileSpec **pval);
```

pval

On return, this parameter contains the address of a pointer to a file specification that defines the target output file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

## Targets

*get\_OutputKind*

---

### get\_OutputKind

This method gets the kind of output generated by building the current target.

```
virtual HRESULT get_OutputKind(  
    ECodeWarriorTargetOutputKind *kind);
```

kind

On return, this parameter contains a pointer to the kind of output.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorTargetOutputKind” on page 331](#)

## ICodeWarriorSubTarget

.This interface provides methods for getting information about a subtarget within the current target.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">get_LinkAgainstOutput</a>	<a href="#">get_Target</a>
---------------------------------------	----------------------------

---

---

### get\_LinkAgainstOutput

This method gets whether to link against the output of the current subtarget.

```
virtual HRESULT get_LinkAgainstOutput(  
    VARIANT_BOOL *pVal);
```

pVal

On return, this parameter contains a pointer to a boolean that is set to `true` if the output of the current subtarget should be linked against or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_Target

This method gets the target object for the current subtarget.

```
virtual HRESULT get_Target(  
    ICodeWarriorTarget **pval);
```

pval

On return, this parameter contains the address of a pointer to the current subproject object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



## ICodeWarriorSubProjectTarget

This interface provides methods for getting information about a target within a subproject contained by the current target.

### Inherited Interfaces

- IDispatch

### Methods

The following methods are available for your use:

<a href="#">get_BuildAgainst</a>	<a href="#">get_Name</a>
<a href="#">get_LinkAgainst</a>	

---

### get\_BuildAgainst

This method gets whether to build against the current target within a subproject within a containing target.

```
virtual HRESULT get_BuildAgainst(  
    VARIANT_BOOL *pval) = 0;
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if this target should be built against or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Targets

### *get\_LinkAgainst*

---

### get\_LinkAgainst

This method gets whether to link against the current target within a subproject within a containing target.

```
virtual HRESULT get_LinkAgainst(  
    VARIANT_BOOL __RPC_FAR *pval) = 0;
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if this target should be linked against or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

### get\_Name

This method gets the name of the current target within a subproject within a containing target.

```
virtual HRESULT get_Name(  
    BSTR *pval) = 0;
```

pval

On return, this parameter contains the name of the target.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Targets Data Types

### ECodeWarriorWhichTargetOptions

The ECodeWarriorWhichTargetOptions enumeration provides constants for which target to use in various operations.

**Table 17.1 ECodeWarriorWhichTargetOptions Enumeration**

Constant	Description
kAllTargets	Apply the operation to only the current target.
kCurrentTarget	Apply the operation to all targets.

### ECodeWarriorTargetOutputKind

The ECodeWarriorTargetOutputKind enumeration provides constants for what kind of output to produce when building a target.

**Table 17.2 ECodeWarriorTargetOutputKind Enumeration**

Constant	Description
kCWOutputNone	Produce no output.
kCWOutputFile	Write the output to a file.
kCWOutputDirectory	Write the output to a directory.



## Targets

*Targets Data Types*

---

# Text

---

This chapter shows how to use the Text API to create and manage text operations in the CodeWarrior IDE.

This chapter contains the following sections:

- [Text API Overview](#)
- [Text API Reference](#)

## Text API Overview

The Text API lets plug-ins work with blocks of text in the CodeWarrior IDE.

## Text API Reference

This section describes the functions contained in the following interface:

- [ICodeWarriorTextEngine](#)

## ICodeWarriorTextEngine

This interface provides methods for working with text within the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Methods

This interface exposes the following methods:

<a href="#">get_HasSelection</a>	<a href="#">get_TextLength</a>
<a href="#">get_LineCount</a>	<a href="#">GetTextForLineRange</a>
<a href="#">GetLineForOffset</a>	<a href="#">GetTextForOffsetRange</a>
<a href="#">GetOffsetForLine</a>	<a href="#">InsertText</a>
<a href="#">get_SelectionEnd</a>	<a href="#">put_SelectionEnd</a>
<a href="#">get_SelectionLineEnd</a>	<a href="#">put_SelectionLineEnd</a>
<a href="#">get_SelectionLineStart</a>	<a href="#">put_SelectionLineStart</a>
<a href="#">get_SelectionStart</a>	<a href="#">put_SelectionStart</a>
<a href="#">get_SelectionText</a>	<a href="#">put_SelectionText</a>

---

### get\_HasSelection

This method gets whether the user has selected a block of text.

```
get_HasSelection(
    VARIANT_BOOL *pval);
```

pval

On return, this parameter contains a pointer to a boolean that is set to `true` if the user has highlighted a block of text or `false` if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_LineCount

This method gets the number of lines of text the user has selected.

```
get_LineCount(  
    int *pval);
```

`pval`

On return, this parameter contains a pointer to an integer that indicates how many lines the user has selected.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## GetLineForOffset

This method gets the line number of a given character in the source file.

```
GetLineForOffset(  
    int offset,  
    int *line);
```

`offset`

The position from the top of the file for which to get a line number.

`line`

On return, this parameter contains a pointer to an integer indicating the line number of the specified character.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Text***GetOffsetForLine*

---

**GetOffsetForLine**

This method gets the number of characters from the top of the source file to the specified line.

```
GetOffsetForLine(  
    int line,  
    int *offset);
```

line

The line number in question.

offset

On return, this parameter contains a pointer to an integer indicating the number of characters to the specified line.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**get\_SelectionEnd**

This method gets the number of characters from the top of the file to the end of the user's selection.

```
get_SelectionEnd(  
    int *pval);
```

pval

On return, this parameter contains a pointer to an integer that indicates how many characters from the top of the file to the end of the user's selection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.



---

## get\_SelectionLineEnd

This method gets how many lines from the top of the source file to the last line of the user's selection.

```
get_SelectionLineEnd(  
    int *pval);
```

pval

On return, this parameter contains a pointer to an integer that indicates how many lines from the top of the file to the last line of the user's selection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_SelectionLineStart

This method gets how many lines from the top of the source file to the first line of the user's selection.

```
get_SelectionLineStart(  
    int *pval);
```

pval

On return, this parameter contains a pointer to an integer that indicates how many lines from the top of the file to the first line of the user's selection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Text***get\_SelectionStart*

---

---

**get\_SelectionStart**

This method gets the number of characters from the top of the file to the start of the user's selection.

```
get_SelectionStart(  
    int *pval);
```

pval

On return, this parameter contains a pointer to an integer that indicates how many characters from the top of the file to the start of the user's selection.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

**get\_SelectionText**

This method gets the text the user has selected.

```
get_SelectionText(  
    BSTR *pval);
```

pval

On return, this parameter contains the text the user has selected.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## get\_TextLength

This method gets the number of characters the user has selected.

```
get_TextLength(  
    int *pval);
```

pval

On return, this parameter contains a pointer to an integer that indicates how many characters the user has selected.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetTextForLineRange

This method gets the text within a specified line range.

```
GetTextForLineRange(  
    int lineStart,  
    int lineEnd,  
    BSTR *pval);
```

lineStart

The first line of the block of text to get.

lineEnd

The last line of the block of text to get.

pval

On return, this parameter contains the text specified by the lineStart and lineEnd parameters.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

**Text***GetTextForOffsetRange*

---

**GetTextForOffsetRange**

This method gets the text within a specified offset from the top of the source file.

```
GetTextForOffsetRange(  
    int selStart,  
    int selEnd,  
    BSTR *pval);
```

`selStart`

The first character of the block of text to get.

`selEnd`

The last character of the block of text to get.

`pval`

On return, this parameter contains the text specified by the `selStart` and `selEnd` parameters.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**InsertText**

This method inserts text at the current position within the source file. You can use [put SelectionStart](#) to position the text insertion point.

```
InsertText(  
    BSTR val);
```

`val`

The text to insert.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## put\_SelectionEnd

This method moves the end of the selection block to the specified position, counted from the top of the source file.

```
put_SelectionEnd(  
    int val);
```

val

The number of characters from the top of the file to the new end of the selection block.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_SelectionLineEnd

This method moves the end of the selection block to a specified line.

```
put_SelectionLineEnd(  
    int val);
```

val

The line number of the new end of the selection block.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## put\_SelectionLineStart

This method moves the start of the selection block to a specified line.

```
put_SelectionLineStart(  
    int val);
```

## Text

### *put\_SelectionStart*

---

`val`

The line number of the new start of the selection block.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

### `put_SelectionStart`

This method moves the start of the selection block to the specified position, counted from the top of the source file.

```
put_SelectionStart(
    int val);
```

`val`

The number of characters from the top of the file to the new start of the selection block.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

### `put_SelectionText`

This method replaces the current selection with a new block of text.

```
put_SelectionText(
    BSTR val);
```

`val`

The new block of text.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

# Toolbar

---

This chapter shows how to use the Toolbar API to create and manage buttons in the CodeWarrior IDE toolbar.

This chapter contains the following sections:

- [Toolbar API Overview](#)
- [Toolbar API Reference](#)

## Toolbar API Overview

The Toolbar API is a set of interfaces that lets a plug-in create and manipulate buttons in the CodeWarrior IDE toolbar. The API uses the standard COM .

## Toolbar API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorCustomToolbarItem](#)
- [ICodeWarriorPopupMenuToolbarItem](#)
- [ICodeWarriorToggleButtonToolbarItem](#)
- [ICodeWarriorToolbar](#)
- [ICodeWarriorToolbarInstanceCreationNotification](#)
- [ICodeWarriorToolbarItemHelp](#)
- [ICodeWarriorToolbarItemRegistry](#)

## ICodeWarriorCustomToolbarItem

This interface exposes methods for creating, drawing, and getting information about a toolbar item.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">CreateItemControl</a>	<a href="#">GetItemRepresentationWidth</a>
<a href="#">DrawItemRepresentation</a>	<a href="#">GetItemSizeInfo</a>

---

### CreateItemControl

This method creates a new item to put in a toolbar.

```
virtual void* CreateItemControl(  
    ICodeWarriorToolbar *inToolbar,  
    void *hwndParent);
```

*inToolbar*

A pointer to the toolbar in which to create the new item.

*hwndParent*

A pointer to the window handle of the parent window.

Returns A pointer to the new item.



---

## DrawItemRepresentation

This method draws the current item in a specified specified graphics context.

```
virtual HRESULT DrawItemRepresentation(  
    void *inGraphicsContext,  
    LONG xPos,  
    LONG yPos,  
    LONG width,  
    LONG height);
```

`inGraphicsContext`

A pointer to the graphics context in which to draw the item.

`xPos`

The X position at which to draw the item.

`yPos`

The Y position at which to draw the item.

`width`

The width of the item.

`height`

The height of the item.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## Toolbar

### *GetItemRepresentationWidth*

---

## GetItemRepresentationWidth

This method gets the width the current item occupies in a given graphics context.

```
virtual LONG GetItemRepresentationWidth(
    void *inGraphicsContext);
```

`inGraphicsContext`

A pointer to the graphics context.

Returns A long integer indicating the width the item occupies in the specified graphics context.

---

## GetItemSizeInfo

This method gets the size of the current item and whether the item can be resized.

```
virtual HRESULT GetItemSizeInfo(
    LONG &outMinWidth,
    BOOL &outResizable);
```

`outMinWidth`

On return, this parameter contains the address of a long integer indicating the size of the item.

`outResizable`

On return, this parameter contains the address of a boolean that is set to `true` if the item can be resized or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorPopupMenuToolbarItem

This interface provides methods for creating and working with popup menus on a toolbar.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">BuildPopupMenuList</a>	<a href="#">GetItemWidth</a>
<a href="#">CleanupPopupMenuList</a>	<a href="#">GetSampleTextString</a>
<a href="#">GetInitialState</a>	<a href="#">HandlePopupMenuSelection</a>

---

## BuildPopupMenuList

This method creates a list of items to put in a popup menu.

```
virtual HRESULT BuildPopupMenuList(  
    ICodeWarriorToolbar *inToolbar,  
    void *inItemData,  
    LONG inKeyboardModifiers,  
    SPopupMenuToolbarItem *&outItems,  
    LONG &outItemCount,  
    LONG &outSelItem)
```

`inToolbar`

A pointer to the toolbar on which to place the popup menu.

`inItemData`

A pointer to the items to put on the menu.

`inKeyboardModifiers`

A pointer to the hotkeys what select items on the menu.

## Toolbar

### CleanupPopupMenuList

---

outItems

On return, this parameter contains a pointer to the address of the items in the menu.

outItemCount

On return, this parameter contains the address of a long indicating how many items are in the menu.

outSelItem

On return, this parameter contains the address of the selected item.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorToolbar" on page 354](#)  
["SPopupMenuToolbarItem" on page 366](#)

---

## CleanupPopupMenuList

This method resets a popup menu's item list.

```
virtual HRESULT CleanupPopupMenuList(
    SPopupMenuToolbarItem *inItems,
    LONG inItemCount);
```

inItems

A pointer to the items to place in the menu.

inItemCount

The number of items the menu should have.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## GetInitialState

This method gets the initial state of the popup menu.

```
virtual HRESULT GetInitialState(  
    ICodeWarriorToolbar *inToolbar,  
    void *inItemData,  
    STR &outCurrentStr,  
    CWToolbarIconInfo& outIcon,  
    BOOL &outIsEnabled)
```

`inToolbar`

A pointer to the toolbar on which the menu appears.

`inItemData`

A pointer to the items to put on the menu.

`outCurrentStr`

On return, this parameter contains the string value of the current item in the popup menu.

`outIcon`

On return, this parameter contains the address of information about the icon associated with the current item in the menu.

`outIsEnabled`

On return, this parameter contains the address of a boolean that is set to `true` if the menu is enabled or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“ICodeWarriorToolbar” on page 354](#)

---

## GetItemWidth

This method gets the width of the current item.

```
virtual HRESULT GetItemWidth(LONG &outWidth);
```

outWidth

On return, this parameter contains the address of a long indicating the width of the current item.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## GetSampleTextString

This method gets a sample text string from the current popup menu.

```
virtual HRESULT GetSampleTextString(  
    BSTR &outSampleStr);
```

outSampleStr

On return, this parameter contains the sample string.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## HandlePopupSelection

The IDE calls this method to let the plug-in know to perform the action associated with a popup on a toolbar.

```
virtual HRESULT HandlePopupSelection(  
    void *inItemData  
    ICodeWarriorToolbar *inToolbar,  
    LONG itemIndex,  
    SPopupMenuToolbarItem *inItems,  
    LONG inItemCount);
```

`inToolbar`

A pointer to the toolbar on which the popup resides.

`inItemData`

A pointer to the items on the toolbar

`itemIndex`

The item number of the item whose event you want to dispatch.

`inItems`

A pointer to the list of choices in the popup.

`inItemCount`

The number of the item (in the list specified by the `inItems` parameter) selected by the user.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorToggleButtonToolbarItem

This API exposes a method that lets you determine if a toggle button on a toolbar has been pressed.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

---

[GetInitialState](#)[StateChanged](#)

---

---

### GetInitialState

This method gets the initial state of a toggle button on a toolbar.

```
virtual HRESULT GetInitialState(  
    ICodeWarriorToolbar *inToolbar,  
    void *inItemData,  
    CWToolbarItemID inItemID,  
    BOOL& outInitialState,  
    BOOL& outEnabled)
```

*inToolbar*

A pointer to the toolbar on which the toggle button appears.

*inItemData*

A pointer to the item data for the toggle button.

*outInitialState*

On return, this parameter contains the address of a boolean set to true if the toggle button is in its selected state (toggle is on) or false if it is in its deselected state (toggle is off).

*outEnabled*

On return, this parameter contains the address of a boolean set to true if the toggle button is enable or false if it is disabled.



Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## StateChanged

This method sets the state of a toggle button to on or off.

```
virtual HRESULT StateChanged(  
    ICodeWarriorToolbar *inToolbar,  
    CWToolbarItemID inItemID,  
    BOOL inNewState);
```

*inToolbar*

A pointer to the toolbar on which the toggle button appears.

*inItemID*

The ID of

*inNewState*

Set this parameter to true to put the toggle button in its selected (on) state or false to set the toggle button in its deselected (off) state.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorToolbar

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">GetContainingDocument</a>	<a href="#">SetToolbarItemEnabled</a>
<a href="#">GetToolbarHeight</a>	<a href="#">SetToolbarItemIcon</a>
<a href="#">GetToolbarItemText</a>	<a href="#">SetToolbarItemText</a>
<a href="#">GetToolbarItemValue</a>	<a href="#">SetToolbarItemValue</a>
<a href="#">IsToolbarVisible</a>	<a href="#">ShowToolbar</a>
<a href="#">ResetToolbarItem</a>	

---

### GetContainingDocument

This method gets the document that contains the toolbar.

```
virtual ICodeWarriorDocumentPrivate*
    GetContainingDocument(void);
```

Returns A pointer to the document that contains the toolbar.

See Also ["ICodeWarriorDocument" on page 160](#)

---

### GetToolbarHeight

This method gets the height of the toolbar.

```
virtual LONG GetToolbarHeight(void);
```

Returns The height of the toolbar.

---

## GetToolbarItemText

This method gets the text label for a specified toolbar item.

```
virtual HRESULT GetToolbarItemText(  
    const CWPluginID inPluginID,  
    const CWToolbarItemID inItemID,  
    BSTR &outItemText);
```

inPluginID

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

inItemID

The toolbar item for which to get the text label.

outItemText

On return, this parameter contains the text of the toolbar item

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## GetToolbarItemValue

This method gets the value of a specified item in the toolbar

```
virtual HRESULT GetToolbarItemValue(  
    const CWPluginID inPluginID,  
    const CWToolbarItemID inItemID,  
    LONG &outValue);
```

inPluginID

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

## Toolbar

### *IsToolbarVisible*

---

`inItemID`

The toolbar item for which to get the value.

`outValue`

On return, this parameter contains the address of a long indicating the value of the toolbar item specified by the `inItemID` parameter.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## IsToolbarVisible

This method gets whether the current toolbar is visible.

```
virtual BOOL IsToolbarVisible() = 0;
```

Returns **true** if the current toolbar is visible or **false** if not.

---

## ResetToolbarItem

This method resets a toolbar item to its original state.

```
virtual HRESULT ResetToolbarItem(
    const CWPluginID inPluginID,
    const CWToolbarItemID inItemID);
```

`inPluginID`

TheGUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inItemID`

The toolbar item to reset.

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## SetToolbarItemEnabled

This method sets whether an item in the toolbar is enabled.

```
virtual HRESULT SetToolbarItemEnabled(  
    const CWPluginID inPluginID,  
    const CWToolbarItemID inItemID,  
    BOOL inIsEnabled);
```

inPluginID

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

inItemID

The toolbar item to enable or disable.

inIsEnabled

Set this parameter to true to enable the toolbar item or false to disable it.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## SetToolbarItemIcon

This method sets the icon for a specified item in the toolbar.

```
virtual HRESULT SetToolbarItemIcon(  
    const CWPluginID inPluginID,  
    const CWToolbarItemID inItemID,
```

## Toolbar

### *SetToolbarItemText*

---

```
const CWToolbarIconInfo inIconData);
```

`inPluginID`

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inItemID`

The toolbar item for which to set icon information.

`inIconData`

The icon data for the icon associated with the specified toolbar item.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## SetToolbarItemText

This method sets the text label for a specified item in the toolbar.

```
virtual HRESULT SetToolbarItemText(
    const CWPluginID inPluginID,
    const CWToolbarItemID inItemID,
    BSTR inNewText);
```

`inPluginID`

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inItemID`

The toolbar item for which to set the text label.

`inNewText`

The text to which to set the text label of the item specified in the `inItemID` parameter.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## SetToolbarItemValue

This method sets the value of a specified item in the toolbar.

```
virtual HRESULT SetToolbarItemValue(  
    const CWPluginID inPluginID,  
    const CWToolbarItemID inItemID,  
    LONG inValue);
```

`inPluginID`

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inItemID`

The toolbar item for which to set the value.

`inValue`

The value to set for the toolbar item specified in `inItemID`.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolbarItemID” on page 366](#)

---

## ShowToolbar

This method sets whether to show the current toolbar.

```
virtual HRESULT ShowToolbar(  
    BOOL inShow) = 0;
```

**Toolbar**  
*ShowToolbar*

---

`inShow`

`true` to show the current toolbar is visible or `false` to not show it.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.



## ICodeWarriorToolbarInstanceCreationNotification

This interface provides methods for determining whether an item has been created or destroyed.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

---

<a href="#">ItemCreated</a>
-----------------------------

---

<a href="#">ItemDestroyed</a>
-------------------------------

---

---

### ItemCreated

This method lets you find out whether a new item has been created on a toolbar.

```
virtual HRESULT ItemCreated(  
    ICodeWarriorToolbar *inToolbar,  
    void *&outItemData);
```

*inToolbar*

A pointer to the toolbar on which to create a new item.

*outItemData*

On return, this parameter contains a pointer to a reference for the new item's information.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Toolbar

### *ItemDestroyed*

---

## ItemDestroyed

This method notifies you when an item has been removed from a toolbar.

```
virtual HRESULT ItemDestroyed(  
    ICodeWarriorToolbar *inToolbar,  
    void *inItemData);
```

*inToolbar*

A pointer to the toolbar you want to be

*inItemData*

A pointer to the destroyed item.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorToolbarItemHelp

This interface provides a way to get the help text for a toolbar item.

### Inherited Interfaces

- IUnknown

The following methods are available for your use:

---

<a href="#">GetHelpString</a>	
-------------------------------	--

---

---

### GetHelpString

This method gets the help string for a specified toolbar item.

```
virtual HRESULT GetHelpString(  
    CWToolbarItemID itemID,  
    BSTR &outHelpString);
```

itemID

The ID of the item for which to get the help string.

outHelpString

On return, this parameter contains the help string.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorToolbarItemRegistry

This interface provides methods for creating registry items for toolbar items and icons.

### Inherited Interfaces

- IUnknown

The following methods are available for your use:

---

[RegisterToolbarIcons](#)[RegisterToolbarItem](#)

---

---

## RegisterToolbarIcons

This method creates a registry entry a toolbar icon.

```
virtual HRESULT RegisterToolbarIcons(  
    const CWPluginID inPluginID,  
    const CWToolbarIconRegistryInfo &inIconData);
```

**inPluginID**

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

**inIconData**

The address of the icon data to register.

**Returns** S\_OK if this method call succeeded or an appropriate error if it failed.

**See Also** [CWToolbarIconRegistryInfo](#)

## RegisterToolBarItem

This method registers an item in a toolbar.

```
virtual HRESULT RegisterToolBarItem(  
    const CWPluginID inPluginID,  
    const CWToolBarItemID inItemID,  
    const long inItemType,  
    const CWToolBarIconInfo inIconData,  
    const BSTR inItemName,  
    IUnknown *itemHandler);
```

`inPluginID`

The GUID for the plug-in. Usually this is the class ID of the main class of your plug-in.

`inItemID`

The ID of the item to register.

`inItemType`

The Type of the item to register.

`inIconData`

The icon data associated with the item.

`inItemName`

The name of the item

`itemHandler`

A pointer to the handler for the item.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [“CWToolBarItemID” on page 366](#)

## Toolbar Data Types

### SPopupMenuToolbarItem

The following enumeration (from CodeWarriorToolbar.h) defines the toolbar items used in the CodeWarrior Toolbar API:

---

```
struct SPopupMenuToolbarItem
{
    unsigned long    itemFlags;
    BSTR            itemText;
    void            *userData;
    CWToolbarIconInfo itemIcon;
};
```

---

### CWToolbarItemID

The following enumeration (from CodeWarriorToolbar.h) defines the toolbar item IDs used in the CodeWarrior Toolbar API:

---

```
typedef long CWToolbarItemID;
enum
{
    tbItemType_Separator,
    tbItemType_CommandButton,
    tbItemType_ToggleButton,
    tbItemType_CheckBox,
    tbItemType_PopupButton,
    tbItemType_PopupList, // Bevel button w/text on MacOS
                          // Combo box on Windows
    tbItemType_Custom
};
```

---

## CWToolbarIconRegistryInfo

The following structure (from CodeWarriorToolbar.h) defines the toolbar registry information used in the CodeWarrior Toolbar API:

---

```
#if defined(macintosh) || defined(_LATITUDE_)
typedef void* CWToolbarIconRegistryInfo;
#elif defined(WIN32)
typedef struct
{
    HBITMAP hotImages;
    HBITMAP normalImages;
    COLORREF maskColor;
} CWToolbarIconRegistryInfo;
#endif
```

---

## Toolbar Constants

### Item Flags

The following item flags are defined in CodeWarriorToolbar.h for use with the CodeWarrior toolbar API:

CWPopup_Checked	1
CWPopup_Disabled	2
CWPopup_Underline	4

---



**Toolbar**  
*Toolbar Constants*

---



# Version Control

---

This chapter shows how to use the Version Control API to work with the Version Control system in the CodeWarrior IDE.

This chapter contains the following sections:

- [Version Control API Reference](#)

## Version Control API Reference

This section describes the functions contained in the following interfaces:

- [ICodeWarriorVersionControl](#)
- [ICodeWarriorVCSState](#)
- [ICodeWarriorVCSFileStateListener](#)

The VCS interfaces make use of various data types, which are described in the following section:

- [VCS Data Types](#)

## ICodeWarriorVersionControl

This interface provides methods for the basic Version Control operations (checking in, checking out, and so on).

### Inherited Interfaces

- IUnknown

This interfaces exposes the following methods:

<a href="#">CheckIn</a>	<a href="#">get_Name</a>
<a href="#">CheckOut</a>	<a href="#">GetVCSSState</a>
<a href="#">Connect</a>	<a href="#">IsConnected</a>
<a href="#">Disconnect</a>	<a href="#">UndoCheckOut</a>
<a href="#">Get</a>	<a href="#">UnLock</a>

---

### CheckIn

This method checks in the files in a specified collection of files.

```
virtual HRESULT CheckIn(
    IFileSpecCollection *fileSpecCollection);
```

fileSpecCollection

A pointer to a collection object containing the files to check in.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

### CheckOut

This method checks out the files in a specified collection of files.

```
virtual HRESULT CheckOut(
```

```
IFileSpecCollection *fileSpecCollection);
```

fileSpecCollection

A pointer to a collection object containing the files to check out.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## Connect

This method connects to the Version Control database.

```
virtual HRESULT Connect(void);
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## Disconnect

This method disconnects from the Version Control database.

```
virtual HRESULT Disconnect(void);
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## Get

This method gets a collection of files from the Version Control database.

```
virtual HRESULT Get(
    IFileSpecCollection *fileSpecCollection);
```

---

## Version Control

### *get\_Name*

---

fileSpecCollection

A pointer to a collection object containing the files toget.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

### get\_Name

This method gets the name of the Version Control system.

```
virtual HRESULT get_Name(
    BSTR *vcsName);
```

vcsName

On return, this parameter contains the name of the Version Control system.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

### GetVCSState

This method gets the Version Control state for a specified file.

```
virtual HRESULT GetVCSState(
    IFileSpec *fileSpec,
    ICodeWarriorVCSState **vcsState);
```

fileSpec

A pointer to the file specification for which to get

vcsState

On return, this parameter contains the address of a pointer to the state of the file specified by the fileSpec parameter.

---

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)

---

## UnLock

This method unlocks the files in a collection of files.

```
virtual HRESULT UnLock(
    IFileSpecCollection *fileSpecCollection);
```

`fileSpecCollection`

A pointer to the collection of files to unlock.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

---

## IsConnected

This method gets whether a connection to the Version Control database. You might want to check the result of this method before calling [Connect](#) or [Disconnect](#).

```
virtual HRESULT IsConnected(
    VARIANT_BOOL *pval);
```

`pval`

On return, this parameter contains a pointer to a boolean that is set to `true` if a connection to the Version Control database exists or `false` if no connection exists.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

---

## UndoCheckOut

This method performs an UndoCheckOut operation (essentially, it restores files to their previous version) on a collection of files. Using it may cause changes to be lost.

```
virtual HRESULT UndoCheckOut(  
    IFileSpecCollection *fileSpecCollection);  
  
fileSpecCollection
```

A pointer to the collection of files on which to perform an UndoCheckOut operation.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“Using the Collections API” on page 77](#)

## ICodeWarriorVCSState

This interface provides methods that let you can use to check on the state of the Version Control system.

### Inherited Interfaces

- IUnknown

### Methods

The interface exposes the following methods:

<a href="#">get_CKIDState</a>	<a href="#">get_DBState</a>
<a href="#">get_FileLockState</a>	

---

### get\_CKIDState

This method gets the Version Control state (checked in, checked out, not in the Version Control system, and so on) for the current file.

```
virtual HRESULT get_CKIDState(
    ECodeWarriorVCSCKIDState *type);
```

type

On return, this parameter contains a pointer to the state of the file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorVCSCKIDState” on page 378](#)

---

## get\_DBState

This method gets the state of the Version Control database.

```
virtual HRESULT get_DBState(
    ECodeWarriorVCSDBState *type);
```

type

On return, this parameter contains a pointer to the state of the database.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorVCSDBState” on page 379](#)

---

## get\_FileLockState

This method gets the lock state for the current file.

```
virtual HRESULT get_FileLockState(
    ECodeWarriorVCSFileLockState *type);
```

type

On return, this parameter contains a pointer to the lock state of the current file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“ECodeWarriorVCSFileLockState” on page 379](#)



## ICodeWarriorVCSFileStateListener

This interface provides a method that lets you monitor the state of a file in the Version Control system.

### Inherited Interfaces

- IUnknown

### Methods

This interface exposes the following methods:

---

<a href="#">StateChanged</a>	
------------------------------	--

---

---

### StateChanged

This method sends a message all listeners of this interface when a file's VCS state has changed.

```
virtual HRESULT StateChanged(  
    IFileSpec *fileSpec,  
    ICodeWarriorVCSState *vcsState);
```

fileSpec

The file specification for the file whose state has changed.

vcsState

The new state of the file.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“IFileSpec” on page 185](#)  
[“ICodeWarriorVCSState” on page 375](#)

## VCS Data Types

The following data types are used with the VCS API:

- [ECodeWarriorVCSCKIDState](#)
- [ECodeWarriorVCSDBState](#)
- [ECodeWarriorVCSFileLockState](#)

### ECodeWarriorVCSCKIDState

This enumeration describes whether a file is in the version control system and its state within the version control system.

---

**NOTE** CKID only has meaning on the Mac OS, so this enumeration has no meaning on any other operating system.

---

**Table 20.1** ECodeWarriorVCSCKIDState Enumeration

Constant	Description
<code>vcSCKIDNotChecked</code>	The file is not in the version control system.
<code>vcSNoCKID</code>	No CKID is available.
<code>vcSCKIDCheckedIn</code>	The file is in the version control system and is currently checked in.
<code>vcSCKIDCheckedOut</code>	The file is in the version control system and is currently checked out.
<code>vcSCKIDMRO</code>	The file is in the version control system and is marked read-only (MRO).

### ECodeWarriorVCSDBState

This enumeration describes the state of a file in the version control database.

**Table 20.2 ECodeWarriorVCSDBState Enumeration**

Constant	Description
vcsDBNotChecked	The file has not been checked for its VCS state. This state can appear in in a dialog box or message window if the version control system was unable to check the state of the file.
vcsDBNotInWorkingDir	The file is not in the working directory defined in the VCS Preference Panel.
vcsDBNotInDatabase	The file is not in the VCS database.
vcsDBCheckedIn	The file is in the VCS database and is checked in
vcsDBCheckedOut	The file is in the VCS data base and is checked out.

### ECodeWarriorVCSFileLockState

This enumeration describes the lock state of a file, from the perspective of the version control system.

**Table 20.3 ECodeWarriorVCSFileLockState Enumeration**

Constant	Description
vcsLockNotChecked	This file is not in the version control system.
vcsVolLocked	The volume in which this file resides is locked (as is the file, by extension).
vcsFileLocked	The file is locked.
vcsFileReadOnly	The file is read-only.



**Version Control**  
*VCS Data Types*

---

# Windows

---

This chapter shows how to use the Windows API to create and manage windows in the CodeWarrior IDE.

This chapter contains the following sections:

- [Windows API Overview](#)
- [Using the Windows API](#)
- [Windows API Reference](#)

## Windows API Overview

The Windows API is a set of interfaces that allows a plug-in to create and manipulate windows in the CodeWarrior IDE. The API uses the standard COM interfaces.

Windows are registered through standard commands (menus) and allow setting of standard window attributes.

Windows events are platform-specific and should be handled accordingly.

## Using the Windows API

You will need to create an `ICodeWarriorMenuManager` interface and a call to `QueryInterface()` to get this interface. Once you have created the window you will attach an event/command handler to the window. The latter is done when registering commands with `RegisterCommand()`.

## Windows API Reference

This section describes the methods contained in the following interfaces:

- [ICodeWarriorWindowManager](#)
- [ICodeWarriorWindow](#)
- [ICodeWarriorWindowEvents](#)

The Windows API interfaces use a data type described in the following section:

- [Windows Data Types](#)

## ICodeWarriorWindowManager

This interface is provided to allow plug-ins to create windows in the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Methods

This interface implements the following methods:

<a href="#">CenterWindow</a>	<a href="#">GetIDEMainWindow</a>
<a href="#">CreateCodeWarriorWindow</a>	<a href="#">IsIDEInMDIMode</a>

---

### CenterWindow

This method centers the selected window over the main window.

This method centers a specified window on the client screen or centers the main window if the IDE is in MDI mode.

```
virtual HRESULT CenterWindow(CWNativeWindowType  
    window, BOOL fIsDialog, int reserved) = 0;
```

window

The CodeWarrior window you want to center on the screen.

fIsDialog

Set this flag to true if the window is a dialog or false otherwise.

reserved

Set this parameter to 0.

## Windows

### CreateCodeWarriorWindow

---

## CreateCodeWarriorWindow

Call this method to create a new window in the CodeWarrior IDE. You can use this interface to access window methods. You will also need an event handler for your window.

```
virtual ICodeWarriorWindow*
    CreateCodeWarriorWindow(
        const CWPluginID inPluginID,
        ULONG inAttributes) = 0;
```

`inPluginID`

The ID for the plug-in.

`inAttributes`

Attributes that describe the type of window you want. Multiple attributes can be set for any window. The following attributes are allowed:

Attributes for all platforms:

<code>CWWindow_CanClose</code>	The window has a close box
<code>CWWindow_CanResize</code>	The window is resizable
<code>CWWindow_Floating</code>	The window floats above all other IDE windows
<code>CWWindow_PutInOpenWindowsMenu</code>	The window name is put in the available windows in the Window menu.
<code>CWWindow_Modal</code>	The window is modal



## Mac OS specific attributes:

CWindow_GetIdleTime	The window event handler's Idle method will be called
CWindow_GetSelectClick	true if a click both brings a window forward and affects its content or false if a click only brings a window forward. This setting works only on the Mac OS.
CWindow_DelaySelect	true to bring a window forward on a mouse-up event or false to bring a window forward on a mouse-down event. This setting works only on the Mac OS.

Returns A pointer to an `ICodeWarriorWindow` object.

See Also [“ICodeWarriorWindow” on page 387](#)

## IsIDEInMDIMode

This method is used to determine if the IDE is in Multiple Document Interface (MDI) mode.

```
virtual BOOL IsIDEInMDIMode()
```

Returns true if window is in IDE MDI Mode or false otherwise. If this method returns false, the IDE is in Floating Document Interface (FDI) mode.

**Windows***GetIDEMainWindow*

---

---

**GetIDEMainWindow**

This method gets a window handle for the IDE's main window.

---

**NOTE** This method works only on Win32.

---

```
virtual HRESULT GetIDEMainWindow(  
    HWND *mainWnd)
```

mainWnd

On return, this parameter contains a pointer to the IDE's main window.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorWindow

This interface is provided to allow plug-ins to manipulate windows in the CodeWarrior IDE.

### Inherited Interfaces

- IUnknown

### Methods

The following methods are available for your use:

<a href="#">AssociateWindowWithProject</a>	<a href="#">SetBackBrushes</a>
<a href="#">CreateToolbar</a>	<a href="#">SetCodeWarriorWindowInitialBounds</a>
<a href="#">DestroyCodeWarriorWindow</a>	<a href="#">SetCodeWarriorWindowMinMaxSize</a>
<a href="#">GetCodeWarriorWindowSizeLocation</a>	<a href="#">SetCodeWarriorWindowTitle</a>
<a href="#">GetNativeWindowReference</a>	<a href="#">SetDialogColors</a>
<a href="#">GetNativeXWindowReference</a>	<a href="#">SetEventHandler</a>
<a href="#">GetWindowToolbar</a>	<a href="#">SetMaximumSleepTime</a>
<a href="#">HasAttribute</a>	<a href="#">SelectCodeWarriorWindow</a>
<a href="#">MoveCodeWarriorWindow</a>	<a href="#">ShowCodeWarriorWindow</a>
<a href="#">PutBehind</a>	<a href="#">UpdatePort</a>
<a href="#">ReorderCodeWarriorWindow</a>	

**NOTE** The [PutBehind](#), [ReorderCodeWarriorWindow](#), [SetBackBrushes](#), [SetDialogColors](#), and [SetMaximumSleepTime](#) methods work only on the Mac OS. The [GetNativeXWindowReference](#) method works only on unix-based systems.

**Windows***AssociateWindowWithProject*

---

---

**AssociateWindowWithProject**

This method associates a window with a project. If the window is in front and the user performs actions (for example, Compile) related to a project, the project associated with the window becomes associated with that window.

```
virtual HRESULT AssociateWindowWithProject(  
    IUnknown *inProjectObject)
```

`inProjectObject`

The project your window is to be associated with.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

**CreateToolbar**

Creates a new toolbar in the window. The IDE will call the event handler's `GetDefaultToolbarItems()`.

```
virtual HRESULT CreateToolbar(  
    BSTR inToolbarTitle,  
    ICodeWarriorToolbar *&outToolbar) = 0;
```

`inToolbarTitle`

The title of the new toolbar.

`outToolbar`

The IDE will provide a pointer to the toolbar interface.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

See Also ["ICodeWarriorToolbar" on page 354](#)

---

## DestroyCodeWarriorWindow

This method closes the current window.

```
virtual HRESULT DestroyCodeWarriorWindow()
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## GetCodeWarriorWindowSizeLocation

This method gets the size and location of the window.

```
virtual HRESULT GetCodeWarriorWindowSizeLocation(  
    SHORT &xPos,  
    SHORT &yPos,  
    SHORT &width,  
    SHORT &height) = 0;
```

xPos

The horizontal position of the upper left-hand corner of the window.

yPos

The vertical position of the upper left-hand corner of the window.

width

The width of the window.

height

The height of the window.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

**Windows***GetNativeWindowReference*

---

---

**GetNativeWindowReference**

This method obtains the window reference for the window, according to the platform the plug-in is running on. Once you have this reference, you can use native OS methods to draw whatever you need.

```
virtual CWNativeWindowType  
    GetNativeWindowReference()
```

Returns A value of type `CWNativeWindowType`, which is either a Windows `HWND` structure, or a Mac OS `WindowPtr` structure, depending on the target platform.

---

**GetNativeXWindowReference**

This method gets a native reference to a window on Unix-based operating systems.

---

**NOTE** This method works only on Unix-based operating systems.

---

```
virtual void * GetNativeXWindowReference(  
    CWNativeXWindowPart part) = 0;
```

part

A value within the range specified by the [CWNativeXWindowPart](#) enumeration.

Returns A pointer to the window reference.

See Also [“CWNativeXWindowPart” on page 406](#)

## GetWindowToolbar

This method gets the toolbar associated with the current window, if any.

```
virtual ICodeWarriorToolbar*  
    GetWindowToolbar()
```

Returns A pointer to an ICodeWarriorToolbar interface.

See Also ["ICodeWarriorToolbar" on page 354](#)

---

## HasAttribute

This method discovers whether the current window has a specified attribute.

```
virtual BOOL HasAttribute(  
    ULONG inAttribute)
```

inAttribute

An unsigned long integer indicating the attribute for which to check..

Returns true if the current window has the specified attribute or false if not.

---

## MoveCodeWarriorWindow

This method repositions the size and location of the window.

```
virtual HRESULT MoveCodeWarriorWindow(  
    SHORT xPos,  
    SHORT yPos,  
    SHORT width,  
    SHORT height,
```

---

```
BOOL refresh) = 0;
```

xPos

The horizontal position of the upper left-hand corner of the window.

yPos

The vertical position of the upper left-hand corner of the window.

width

The width of the window.

height

The height of the window.

refresh

Set this parameter to `true` to generate an update event for the content of the window. Set it to `false` to leave the content of the window unchanged.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## PutBehind

This method places the current window behind another window.

**NOTE** This method works only on the Mac OS and Windows.

---

```
virtual HRESULT PutBehind(  
    CWNativeWindowType inBehindWindow);
```

inBehindWindow

The window to place the current window behind. This parameter must be either a Windows `HWND` structure or a Mac OS `WindowPtr` structure, depending on the target



platform.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## ReorderCodeWarriorWindow

This method reorders the current window.

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT ReorderCodeWarriorWindow(  
    ULONG reorderType) = 0;
```

reorderType

An unsigned long integer indicating the type of reordering operation to perform. See the Mac programmer's documentation for more detail.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SetBackBrushes

This method sets the back brush colors for windows for appearance-savvy versions of the Mac OS.

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT SetBackBrushes(  
    ThemeBrush inActiveBackBrush,  
    ThemeBrush inInactiveBackBrush)
```

inActiveBackBrush

Describes the appearance of a window while it is in the

---

## Windows

### *SetCodeWarriorWindowInitialBounds*

---

background. See the Mac Toolbox `ThemeBrush` constants for more information.

`inInactiveBackBrush`

Describes the state of the window when it is in front. See the Mac Toolbox `ThemeBrush` constants for more information.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## SetCodeWarriorWindowInitialBounds

This method sets the initial size and position of the window when it is displayed.

```
virtual HRESULT SetCodeWarriorWindowInitialBounds(
    SHORT xPos,
    SHORT yPos,
    SHORT width,
    SHORT height)
```

`xPos`

The horizontal position of the window, in pixels.

`yPos`

The vertical position of the window, in pixels.

`width`

The width of the window, in pixels.

`height`

The height of the window, in pixels.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

## SetCodeWarriorWindowMinMaxSize

This method sets the minimum and maximum allowable sizes for a window. Users will not be able to resize the window smaller than the minimum size or larger than the maximum size.

```
virtual HRESULT SetCodeWarriorWindowMinMaxSize(  
    SHORT minWidth,  
    SHORT maxWidth,  
    SHORT minHeight,  
    SHORT maxHeight) = 0;
```

`minWidth`

The minimum width of the window, in pixels.

`maxWidth`

The maximum width of the window, in pixels.

`minHeight`

The minimum height of the window, in pixels.

`maxHeight`

The maximum height of the window, in pixels.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## SetCodeWarriorWindowTitle

This method sets the title of the window.

```
virtual HRESULT SetCodeWarriorWindowTitle(  
    BSTR newTitle)
```

`newTitle`

The new window title.

## Windows

### *SetDialogColors*

---

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SetDialogColors

This method sets the dialog colors for windows for non-appearance-savvy versions of the Mac OS.

---

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT SetDialogColors()
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SetEventHandler

This method sets the event handler for the window. The event handler is called for each event the window receives.

```
virtual HRESULT SetEventHandler(
    IUnknown *inEventHandler)
```

*\*inEventHandler*

The event handler for the window. This should be an existing `ICodeWarriorWindowEvents` object or existing `ICodeWarriorCommandHandler` object.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SetMaximumSleepTime

This method sets the maximum sleep time for the current window.

---

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT SetMaximumSleepTime(  
    ULONG inSleepTime) = 0;
```

inSleepTime

An unsigned long specifying the maximum sleep time for the current window. The unit of time is a Mac OS “tick” (1/60 of a second).

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## SelectCodeWarriorWindow

This method selects the window and brings it to the front, if necessary.

```
virtual HRESULT SelectCodeWarriorWindow()
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## ShowCodeWarriorWindow

This method shows or hides the window.

```
virtual HRESULT ShowCodeWarriorWindow(BOOL  
    visible) = 0;
```

visible

Set this field to true if you want the IDE to make the window visible or false if not.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## UpdatePort

This method immediately updates the contents of the window that would normally be updated during the window's next update event.

---

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT UpdatePort() = 0;
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## ICodeWarriorWindowEvents

This interface is provided to allow plug-ins to respond to window events.

### Inherited Interfaces

- IUnknown

### Methods

The following methods apply to windows on all platforms:

<a href="#">ActivateEvent</a>	<a href="#">OkToClose</a>
<a href="#">AdjustCursor</a>	<a href="#">PreBeginUpdate</a>
<a href="#">DeactivateEvent</a>	<a href="#">ToolbarSizeChange</a>
<a href="#">GetDefaultToolbarItems</a>	<a href="#">UpdateEvent</a>
<a href="#">Idle</a>	<a href="#">WindowDestroyed</a>
<a href="#">KeyDownEvent</a>	<a href="#">WindowResizedBy</a>
<a href="#">MouseDownEvent</a>	

**NOTE** The [AdjustCursor](#), [Idle](#), [KeyDownEvent](#), [MouseDownEvent](#), [PreBeginUpdate](#), [UpdateEvent](#), and [WindowResizedBy](#) methods work only on the Mac OS.

## ActivateEvent

This method activates window events.

```
virtual HRESULT ActivateEvent()
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

See Also [“DeactivateEvent” on page 400](#)

---

## AdjustCursor

The method adjusts a cursor over a window on Mac OS.

```
virtual HRESULT AdjustCursor(  
    Point inPortPt,  
    const EventRecord&inMacEvent)
```

`inPortPt`

Specifies the location of the mouse in a window.

`EventRecord`

Specifies the event that occurred in the window. See the Mac OS Toolbox `EventRecord` structure for more information.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## DeactivateEvent

This method deactivates window events.

```
virtual HRESULT DeactivateEvent()
```

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

See Also [ActivateEvent](#)

---

## GetDefaultToolbarItems

This method gets the default toolbar items for a window.

```
virtual HRESULT GetDefaultToolbarItems(  
    const SDefaultToolbarItemInfo *&items,  
    long &itemCount) = 0;
```



items

The registered toolbar items.

itemCount

The number of toolbar items.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## Idle

This method handles idle events for a window on the Mac OS.

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT Idle(  
    const EventRecord &idleEvent);
```

idleEvent

A Mac OS event record. See the Mac OS Toolbox `EventRecord` structure for more information.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## KeyDownEvent

This method handles key down events for window on Mac OS.

**NOTE** This method works only on the Mac OS.

---

```
virtual BOOL KeyDownEvent(  
    const EventRecord &keyEvent)
```

---

## Windows

### MouseDownEvent

---

keyEvent

This field indicates the key event that was received by the window. See the Mac OS Toolbox `EventRecord` structure for more information.

Returns `true` if the event was handled successfully, otherwise `false`.

---

## MouseDownEvent

This method handles a mouse down event within a window on the Mac OS.

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT STDMETHODCALLTYPE
    MouseDownEvent(const EventRecord &mouseEvent,
        BOOL &delaySelect)
```

mouseEvent

This field indicates the type of mouse event received within a window. See the Mac OS Toolbox `EventRecord` structure for more information.

delaySelect

`true` to have the window be selected after a mouse-down event or `false` if not.

Returns `S_OK` if this method call succeeded or an appropriate error if it failed.

---

## OkToClose

The CodeWarrior IDE calls this method to determine whether it is OK to close the window. This method can be used to determine if changes should be saved or not when closing a window.

```
virtual BOOL OkToClose()
```

Returns true if it is OK to close the window or false if not.

---

## PreBeginUpdate

The IDE calls this method to inform a plug-in that the IDE is about to update the window.

---

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT PreBeginUpdate() = 0;
```

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

## ToolbarSizeChange

This method changes the height of the toolbar for the window.

```
virtual HRESULT ToolbarSizeChange(  
    SHORT inNewHeight) = 0;
```

inNewHeight

The new height for the toolbar.

Returns S\_OK if this method call succeeded or an appropriate error if it failed.

---

---

## UpdateEvent

This method update an event in a window on the Mac OS.

---

**NOTE** This method works only on the Mac OS.

---

```
virtual HRESULT UpdateEvent(  
    const RgnHandle updateRgn) = 0;
```

updateRgn

A window area specified by the `updateRgn` field. See the Mac OS Toolbox `RgnHandle` structure for more information.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## WindowDestroyed

The CodeWarrior IDE calls this method when the window handle structure has been destroyed. General cleanup and memory release is done within this method.

```
virtual HRESULT WindowDestroyed() = 0;
```

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

---

## WindowResizedBy

This method specifies how much a window's dimension has changed after a resize.

```
virtual HRESULT WindowResizedBy(  
    SHORT inDeltaH,  
    SHORT inDeltaV)
```

`inDeltaH`

The change in horizontal direction from initial state to final state.

`inDeltaV`

The change in vertical direction from initial state to final state.

Returns **S\_OK** if this method call succeeded or an appropriate error if it failed.

## Windows Data Types

The following data types are used with the Windows API:

- [CWNativeXWindowPart](#)

### CWNativeXWindowPart

This enumeration describes the type of a user tree.

**Table 21.1 CWNativeXWindowPart Enumeration**

Constant	Description
CW_X_WINDOW	The window.
CW_X_DRAWABLE	The drawable part of the window.

# CodeWarrior IDE Interface Definition Language (IDL)

---

This appendix contains the IDL for the CodeWarrior COM API.

```
// 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;
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;
```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

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;

```



```

[
  uuid(5EC306A0-283D-11D0-989C-0080C74ADF8C),
  version(1.1),
  helpstring("Metrowerks CodeWarrior IDE")
]
library CodeWarrior
{
  // TLib : OLE Automation : {00020430-0000-0000-C000-
000000000046}
  importlib("stdole2.tlb");

  [
    odl,
    uuid(110C62F0-CD3C-11D0-846D-00805F3E911D),
    dual,
    oleautomation
  ]
  interface ICodeWarriorProject : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000009), propget]
    HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
    [id(0x00000067), propget]
    HRESULT _stdcall Designs([out, retval]
ICodeWarriorDesignCollection** pval);
    [id(0x00000068), propget]
    HRESULT _stdcall Targets([out, retval]
ICodeWarriorTargetCollection** pval);
    [id(0x00000005), propget]
    HRESULT _stdcall Application([out, retval]
ICodeWarriorApp** val);
    [id(0x00000070), propget]
    HRESULT _stdcall IsVisible([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000075), propget]
    HRESULT _stdcall VersionControl([out, retval]
ICodeWarriorVersionControl** VersionControl);
    [id(0x00000064)]
    HRESULT _stdcall Close();
    [id(0x00000077)]
    HRESULT _stdcall Export([in] BSTR filePath);
  }
}

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
    [id(0x00000078)]
    HRESULT _stdcall ExportByFileSpec([in] IFileSpec*
FileSpec);
    [id(0x00000065)]
    HRESULT _stdcall RemoveBinaries();
    [id(0x00000066)]
    HRESULT _stdcall SetCurrentTarget([in] BSTR targetName);
    [id(0x00000069)]
    HRESULT _stdcall CreateDesign(
        [in] BSTR designName,
        [out, retval] ICodeWarriorDesign** Design);
    [id(0x0000006a)]
    HRESULT _stdcall CreateTarget(
        [in] BSTR targetName,
        [in] BSTR linkerName,
        [in] ICodeWarriorDesign* Design,
        [out, retval] ICodeWarriorTarget** Target);
    [id(0x0000006b)]
    HRESULT _stdcall RemoveTarget([in] ICodeWarriorTarget*
Target);
    [id(0x0000006c)]
    HRESULT _stdcall FindDesign(
        [in] BSTR Name,
        [out, retval] ICodeWarriorDesign** Design);
    [id(0x0000006d)]
    HRESULT _stdcall FindTarget(
        [in] BSTR Name,
        [out, retval] ICodeWarriorTarget** Target);
    [id(0x0000006e)]
    HRESULT _stdcall CloneTarget(
        [in] ICodeWarriorTarget* srcTarget,
        [in] ICodeWarriorProject* srcProject,
        [in] BSTR inDestTargetName,
        [in] VARIANT_BOOL fCopyFileList,
        [in] VARIANT_BOOL fCopyTargetSettings,
        [in] ICodeWarriorDesign* Design,
        [out] ICodeWarriorTarget** outTarget);
    [id(0x0000006f)]
    HRESULT _stdcall GetCurrentTarget([out, retval]
ICodeWarriorTarget** Target);
    [id(0x00000071)]
    HRESULT _stdcall Build([out, retval] long* cookie);
```

```

[id(0x00000072)]
HRESULT _stdcall RemoveDesignByName(
    [in] BSTR designName,
    [in] VARIANT_BOOL fDeleteContainedDesigns);
[id(0x00000073)]
HRESULT _stdcall RemoveDesign(
    [in] ICodeWarriorDesign* Design,
    [in] VARIANT_BOOL fDeleteContainedDesigns);
[id(0x00000074)]
HRESULT _stdcall ReportMessage(
    [in] EReportMsgType msgType,
    [in] BSTR message);
[id(0x00000076)]
HRESULT _stdcall SynchronizeStatus();
[id(0x00000079)]
HRESULT _stdcall BuildAndWaitToComplete([out, retval]
ICodeWarriorBuildMessages** buildMessages);
[id(0x0000007a)]
HRESULT _stdcall GetNamedPluginData(
    [in] BSTR resourceName,
    [in] EPluginDataStorageLoc storeIn,
    [out] IStream** pluginData);
[id(0x0000007b)]
HRESULT _stdcall SetNamedPluginData(
    [in] BSTR resourceName,
    [in] EPluginDataStorageLoc storeIn,
    [in] IStream* pluginData);
[id(0x0000007c)]
HRESULT _stdcall RemoveNamedPluginData(
    [in] BSTR resourceName,
    [in] EPluginDataStorageLoc storeIn);
};

[
    odl,
    uuid(229924D2-FA29-11D1-B330-0060081C5489),
    dual,
    oleautomation
]
interface IFileSpec : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        [id(0x00000001), propput]
        HRESULT _stdcall Name([in] BSTR pval);
        [id(0x00000064), propget]
        HRESULT _stdcall FullPath([out, retval] BSTR* path);
        [id(0x00000064), propput]
        HRESULT _stdcall FullPath([in] BSTR path);
        [id(0x00000065)]
        HRESULT _stdcall Copy([in] IFileSpec* inSpec);
        [id(0x00000066)]
        HRESULT _stdcall Clone([out] IFileSpec** pval);
};

[
    odl,
    uuid(C694D140-95C4-11D1-B31B-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorDesignCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorDesign** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorDesign* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorDesign* pval);
};

[
    odl,
    uuid(4B0EF0A0-95C5-11D1-B31B-0060081C5489),
    dual,
    oleautomation
]

```

```

interface ICodeWarriorDesign : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000001), propput]
    HRESULT _stdcall Name([in] BSTR pval);
    [id(0x00000064), propget]
    HRESULT _stdcall DataModel([out, retval] IUnknown** pval);
    [id(0x00000065), propget]
    HRESULT _stdcall Project([out, retval]
ICodeWarriorProject** pval);
    [id(0x00000066), propget]
    HRESULT _stdcall Targets([out, retval]
ICodeWarriorTargetCollection** pval);
    [id(0x0000006b), propget]
    HRESULT _stdcall BrowserDB([out, retval]
ICodeWarriorSymbolContainer** pval);
    [id(0x00000067)]
    HRESULT _stdcall AddFile(
        [in] BSTR path,
        [in] BSTR groupPath,
        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x00000068)]
    HRESULT _stdcall AddFileByFileSpec(
        [in] IFileSpec* FileSpec,
        [in] BSTR groupPath,
        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x0000006e)]
    HRESULT _stdcall FindAndAddFile(
        [in] BSTR path,
        [in] BSTR groupPath,
        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x00000069)]
    HRESULT _stdcall ContainsTarget(ICodeWarriorTarget*
Target);
    [id(0x0000006a)]
    HRESULT _stdcall
RemoveTargetFromDesign(ICodeWarriorTarget* Target);
    [id(0x0000006f)]
    HRESULT _stdcall CompileFiles(

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        [in] ICodeWarriorProjectFileCollection*
collection,
        [out, retval] long* cookie);
    [id(0x0000006c)]
    HRESULT _stdcall AddAttachment(GUID* attachmentCLSID);
    [id(0x0000006d)]
    HRESULT _stdcall RemoveAttachment(GUID* attachmentCLSID);
};

[
    odl,
    uuid(5976F990-F99B-11D1-B330-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorTargetCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorTarget** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorTarget* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorTarget* pval);
};

[
    odl,
    uuid(F094A000-F996-11D1-B330-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorTarget : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);

```

```

        [id(0x00000001), propput]
        HRESULT _stdcall Name([in] BSTR pval);
        [id(0x00000064), propget]
        HRESULT _stdcall Project([out, retval]
ICodeWarriorProject** Project);
        [id(0x00000065), propget]
        HRESULT _stdcall Design([out, retval] ICodeWarriorDesign**
Design);
        [id(0x0000007a), propget]
        HRESULT _stdcall BrowserEnabled([out, retval]
VARIANT_BOOL* fEnabled);
        [id(0x0000007a), propput]
        HRESULT _stdcall BrowserEnabled([in] VARIANT_BOOL
fEnabled);
        [id(0x00000066), propget]
        HRESULT _stdcall BrowserDB([out, retval]
ICodeWarriorSymbolContainer** catalog);
        [id(0x00000070), propget]
        HRESULT _stdcall ProjectFileCollection([out, retval]
ICodeWarriorProjectFileCollection** ProjectFileCollection);
        [id(0x00000069), propget]
        HRESULT _stdcall TargetFileCollection([out, retval]
ICodeWarriorTargetFileCollection** TargetFileCollection);
        [id(0x0000006a), propget]
        HRESULT _stdcall AccessPaths([out, retval]
ICodeWarriorAccessPaths** pval);
        [id(0x0000006b), propget]
        HRESULT _stdcall UserTrees([out, retval]
ICodeWarriorUserTreeCollection** pval);
        [id(0x00000072), propget]
        HRESULT _stdcall SubTargets([out, retval]
ICodeWarriorSubTargetCollection** subTargetList);
        [id(0x00000067)]
        HRESULT _stdcall AddFile(
                [in] BSTR path,
                [in] BSTR groupPath,
                [out, retval] ICodeWarriorProjectFile**
projectFile);
        [id(0x00000068)]
        HRESULT _stdcall AddFileByFileSpec(
                [in] IFileSpec* FileSpec,
                [in] BSTR groupPath,

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x0000007d)]
    HRESULT _stdcall AddFileByFileSpecCollection(
        [in] IFileSpecCollection* inCollection,
        [in] BSTR groupPath,
        [out] int* pFilesAdded);
    [id(0x0000006d)]
    HRESULT _stdcall FindAndAddFile(
        [in] BSTR path,
        [in] BSTR groupPath,
        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x0000007c)]
    HRESULT _stdcall FindAndAddFileByCollection(
        [in] IBSTRCollection* inCollection,
        [in] BSTR groupPath,
        [out] int* pFilesAdded);
    [id(0x0000006c)]
    HRESULT _stdcall AddSubTarget(
        [in] ICodeWarriorTarget* Target,
        [in] VARIANT_BOOL LinkAgainstOutput);
    [id(0x0000006e)]
    HRESULT _stdcall SetupDebugging([in] VARIANT_BOOL
inTurnOn);
    [id(0x0000006f)]
    HRESULT _stdcall CompileFiles(
        [in] ICodeWarriorProjectFileCollection*
collection,
        [out, retval] long* cookie);
    [id(0x00000071)]
    HRESULT _stdcall GetTargetFileForProjectFile(
        [in] ICodeWarriorProjectFile* projectFile,
        [out, retval] ICodeWarriorTargetFile**
targetFile);
    [id(0x00000073)]
    HRESULT _stdcall GetProjectFileFromFileSpec(
        [in] IFileSpec* FileSpec,
        [out, retval] ICodeWarriorProjectFile**
projectFile);
    [id(0x00000074)]
    HRESULT _stdcall GetNamedPluginData(

```



```

        [in] BSTR resourceName,
        [in] EPluginDataStorageLoc storeIn,
        [out] IStream** pluginData);
[id(0x00000075)]
HRESULT _stdcall SetNamedPluginData(
        [in] BSTR resourceName,
        [in] EPluginDataStorageLoc storeIn,
        [in] IStream* pluginData);
[id(0x00000077)]
HRESULT _stdcall RemoveNamedPluginData(
        [in] BSTR resourceName,
        [in] EPluginDataStorageLoc storeIn);
[id(0x00000076)]
HRESULT _stdcall Build([out, retval] long* cookie);
[id(0x00000078)]
HRESULT _stdcall SynchronizeStatus();
[id(0x00000079)]
HRESULT _stdcall RemoveObjectCode([in] VARIANT_BOOL
deleteDataFiles);
[id(0x0000007b)]
HRESULT _stdcall BuildAndWaitToComplete([out, retval]
ICodeWarriorBuildMessages** buildMessages);
[id(0x0000007e)]
HRESULT _stdcall GetTargetOutput([out, retval]
ICodeWarriorTargetOutput** targetOutput);
[id(0x0000007f)]
HRESULT _stdcall CompileFilesAndWaitToComplete(
collection,
        [out, retval] ICodeWarriorBuildMessages**
buildMessages);
};

[
    odl,
    uuid(F385EEA1-048E-11D2-80C4-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorSymbolContainer : IDispatch {
    [id(0x00000064), propget]

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
    HRESULT _stdcall Target([out, retval] ICodeWarriorTarget**
pval);
    [id(0x00000065), propget]
    HRESULT _stdcall ClassList([out, retval]
ICodeWarriorClassCollection** pval);
    [id(0x00000066)]
    HRESULT _stdcall FindClass(
        [in] BSTR inClassName,
        [out, retval] ICodeWarriorClass**
outClass);
    [id(0x00000067)]
    HRESULT _stdcall FindClassInFile(
        [in] BSTR inClassName,
        [in] IFileSpec* inSpec,
        [out, retval] ICodeWarriorClass**
outClass);
    [id(0x00000068)]
    HRESULT _stdcall AddComponentAttachment([in] GUID*
attachmentCLSID);
    [id(0x00000069)]
    HRESULT _stdcall RemoveComponentAttachment([in] GUID*
attachmentCLSID);
    [id(0x0000006a)]
    HRESULT _stdcall ShowSymbolDeclaration(
        [in] ICodeWarriorSymbol* inSymbol,
        [in] long inForEditing,
        [in] ECodeWarriorShowSymbolLocation
inLocation);
    [id(0x0000006b)]
    HRESULT _stdcall ShowSymbolDefinition(
        [in] ICodeWarriorSymbol* inSymbol,
        [in] long inForEditing,
        [in] ECodeWarriorShowSymbolLocation
inLocation);
};

[
    odl,
    uuid(E98527B0-258E-11D2-80E6-006008C3EEF1),
    dual,
    oleautomation
]
```

---

```

interface ICodeWarriorClassCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorClass** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorClass* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorClass* pval);
};

[
    odl,
    uuid(2F48B6D2-052A-11D2-80C5-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorClass : ICodeWarriorSymbol {
    [id(0x000000c8), propget]
    HRESULT _stdcall BaseClasses([out, retval]
ICodeWarriorBaseClassCollection** pval);
    [id(0x000000c9), propget]
    HRESULT _stdcall SubClasses([out, retval]
ICodeWarriorClassCollection** pval);
    [id(0x000000ca), propget]
    HRESULT _stdcall IsPublic([out, retval] long* pval);
    [id(0x000000cb), propget]
    HRESULT _stdcall IsAbstract([out, retval] long* pval);
    [id(0x000000cc), propget]
    HRESULT _stdcall IsFinal([out, retval] long* pval);
    [id(0x000000cd)]
    HRESULT _stdcall GetDataMembers(
        [in] long inIncludeInherited,
        [out, retval]
ICodeWarriorDataMemberCollection** pval);
    [id(0x000000ce)]
    HRESULT _stdcall GetDataMembersWithAccess(
        [in] long inIncludeInherited,

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        [in] ECodeWarriorAccess inAccessMask,
        [out, retval]
ICodeWarriorDataMemberCollection** pval);
    [id(0x000000cf)]
    HRESULT _stdcall FindDataMemberByName(
        [in] BSTR inName,
        [out, retval] ICodeWarriorDataMember**
pval);
    [id(0x000000d0)]
    HRESULT _stdcall GetMethods(
        [in] long inIncludeInherited,
        [out, retval]
ICodeWarriorMethodCollection** pval);
    [id(0x000000d1)]
    HRESULT _stdcall GetMethodsWithAccess(
        [in] long inIncludeInherited,
        [in] ECodeWarriorAccess inAccessMask,
        [out, retval]
ICodeWarriorMethodCollection** pval);
    [id(0x000000d2)]
    HRESULT _stdcall FindMethodByName(
        [in] BSTR inName,
        [out, retval] ICodeWarriorMethod** pval);
};

[
    odl,
    uuid(2F48B6D0-052A-11D2-80C5-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorSymbol : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall Container([out, retval]
ICodeWarriorSymbolContainer** pval);
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000065), propget]
    HRESULT _stdcall SimpleName([out, retval] BSTR* pval);
    [id(0x00000066), propget]
    HRESULT _stdcall Class([out, retval] ICodeWarriorClass**
pval);

```



```
        [id(0x00000068), propget]
        HRESULT _stdcall DeclarationLocation([out, retval]
ICodeWarriorSourceContext** pval);
        [id(0x00000067), propget]
        HRESULT _stdcall DefinitionLocation([out, retval]
ICodeWarriorSourceContext** pval);
    };

    [
        odl,
        uuid(F385EEA0-048E-11D2-80C4-006008C3EEF1),
        dual,
        oleautomation
    ]
interface ICodeWarriorSourceContext : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
    [id(0x00000064), propput]
    HRESULT _stdcall FileSpec([in] IFileSpec* pval);
    [id(0x00000065), propget]
    HRESULT _stdcall StartOffset([out, retval] long* pval);
    [id(0x00000065), propput]
    HRESULT _stdcall StartOffset([in] long pval);
    [id(0x00000066), propget]
    HRESULT _stdcall EndOffset([out, retval] long* pval);
    [id(0x00000066), propput]
    HRESULT _stdcall EndOffset([in] long pval);
    [id(0x00000067), propget]
    HRESULT _stdcall IsDefined([out, retval] long* pval);
};

    [
        odl,
        uuid(2F48B6D6-052A-11D2-80C5-006008C3EEF1),
        dual,
        oleautomation
    ]
interface ICodeWarriorBaseClassCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
};
```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        [id(00000000)]
        HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorBaseClassInfo**
pval);
        [id(0x00000007)]
        HRESULT _stdcall Add([in] ICodeWarriorBaseClassInfo*
pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in] ICodeWarriorBaseClassInfo*
pval);
    };

    [
        odl,
        uuid(2F48B6D7-052A-11D2-80C5-006008C3EEF1),
        dual,
        oleautomation
    ]
    interface ICodeWarriorBaseClassInfo : IDispatch {
        [id(0x00000064), propget]
        HRESULT _stdcall BaseClass([out, retval]
ICodeWarriorClass** pval);
        [id(0x00000065), propget]
        HRESULT _stdcall Access([out, retval] ECodeWarriorAccess*
pval);
        [id(0x00000066), propget]
        HRESULT _stdcall IsVirtual([out, retval] long* pval);
    };

    typedef [uuid(2F48B6D1-052A-11D2-80C5-006008C3EEF1)]
    enum {
        kAccessNone = 0,
        kPublicAccess = 1,
        kProtectedAccess = 2,
        kPrivateAccess = 4,
        kAccessAll = 65535
    } ECodeWarriorAccess;

    [
        odl,
        uuid(E98527B1-258E-11D2-80E6-006008C3EEF1),

```

```

    dual,
    oleautomation
]
interface ICodeWarriorDataMemberCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorDataMember**
pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorDataMember* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorDataMember*
pval);
};

[
    odl,
    uuid(2F48B6D3-052A-11D2-80C5-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorDataMember : ICodeWarriorSymbol {
    [id(0x000000c8), propget]
    HRESULT _stdcall Access([out, retval] ECodeWarriorAccess*
pval);
    [id(0x000000c9), propget]
    HRESULT _stdcall IsStatic([out, retval] long* pval);
    [id(0x000000ca), propget]
    HRESULT _stdcall IsFinal([out, retval] long* pval);
    [id(0x000000cb), propget]
    HRESULT _stdcall IsTransient([out, retval] long* pval);
    [id(0x000000cc), propget]
    HRESULT _stdcall IsVolatile([out, retval] long* pval);
};

[
    odl,

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

    uuid(E98527B2-258E-11D2-80E6-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorMethodCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorMethod** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorMethod* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorMethod* pval);
};

[
    odl,
    uuid(2F48B6D4-052A-11D2-80C5-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorMethod : ICodeWarriorSymbol {
    [id(0x000000c8), propget]
    HRESULT _stdcall IsConstructor([out, retval] long* pval);
    [id(0x000000c9), propget]
    HRESULT _stdcall IsDestructor([out, retval] long* pval);
    [id(0x000000ca), propget]
    HRESULT _stdcall Access([out, retval] ECodeWarriorAccess*
pval);
    [id(0x000000cb), propget]
    HRESULT _stdcall IsVirtual([out, retval] long* pval);
    [id(0x000000cc), propget]
    HRESULT _stdcall IsAbstract([out, retval] long* pval);
    [id(0x000000cd), propget]
    HRESULT _stdcall IsStatic([out, retval] long* pval);
    [id(0x000000ce), propget]
    HRESULT _stdcall IsInline([out, retval] long* pval);
    [id(0x000000cf), propget]

```



```

        HRESULT _stdcall IsConst([out, retval] long* pval);
        [id(0x000000d0), propget]
        HRESULT _stdcall IsNative([out, retval] long* pval);
        [id(0x000000d1), propget]
        HRESULT _stdcall IsSynchronized([out, retval] long* pval);
};

typedef [uuid(DC953130-943B-11D2-8183-006008C3EEF1)]
enum {
    kShowInEditor = 0,
    kShowInBrowser = 1,
    kUsePreferenceToShow = 2
} ECodeWarriorShowSymbolLocation;

[
    odl,
    uuid(76624FA0-7987-11D2-B361-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorProjectFileCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorProjectFile**
pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorProjectFile* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorProjectFile*
pval);
};

[
    odl,

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

    uuid(59846760-7986-11D2-B361-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorProjectFile : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000064), propget]
    HRESULT _stdcall Project([out, retval]
ICodeWarriorProject** pval);
    [id(0x00000009), propget]
    HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
    [id(0x00000065), propget]
    HRESULT _stdcall Targets([out, retval]
ICodeWarriorTargetCollection** pval);
    [id(0x00000066), propget]
    HRESULT _stdcall VCSState([out, retval]
ICodeWarriorVCSState** pval);
    [id(0x00000067)]
    HRESULT _stdcall CheckOut();
    [id(0x00000068)]
    HRESULT _stdcall CheckIn();
};

[
    odl,
    uuid(9DD0D0B6-ABDA-11D2-9AC2-00C04F79DE48)
]
interface ICodeWarriorVCSState : IDispatch {
    [propget]
    HRESULT _stdcall FileLockState([out, retval]
ECodeWarriorVCSFileLockState* type);
    [propget]
    HRESULT _stdcall CKIDState([out, retval]
ECodeWarriorVCSCKIDState* type);
    [propget]
    HRESULT _stdcall DBState([out, retval]
ECodeWarriorVCSDBState* type);
};

typedef enum {
    vcsLockNotChecked = 0,

```

```

        vcsVolLocked = 1,
        vcsFileLocked = 2,
        vcsFileReadOnly = 3,
        vcsFileReadWrite = 4
    } ECodeWarriorVCSFileLockState;

typedef enum {
    vcsCKIDNotChecked = 0,
    vcsNoCKID = 1,
    vcsCKIDCheckedIn = 2,
    vcsCKIDCheckedOut = 3,
    vcsCKIDMRO = 4
} ECodeWarriorVCSCKIDState;

typedef enum {
    vcsDBNotChecked = 0,
    vcsDBNotInWorkingDir = 1,
    vcsDBNotInDatabase = 2,
    vcsDBCheckedIn = 3,
    vcsDBCheckedOut = 4
} ECodeWarriorVCSDBState;

[
    odl,
    uuid(E14C280E-6799-11D2-9A80-00C04F79DE48),
    dual,
    oleautomation
]
interface ICodeWarriorTargetFileCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorTargetFile**
pval);
    [id(0x00000007)]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        HRESULT _stdcall Add([in] ICodeWarriorTargetFile* pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in] ICodeWarriorTargetFile*
pval);
    };

    [
        odl,
        uuid(3D452250-14EA-11D2-B33B-0060081C5489),
        dual,
        oleautomation
    ]
    interface ICodeWarriorTargetFile : IDispatch {
        [id(0x00000001), propget]
        HRESULT _stdcall Name([out, retval] BSTR* pval);
        [id(0x00000064), propget]
        HRESULT _stdcall Target([out, retval] ICodeWarriorTarget**
pval);
        [id(0x00000009), propget]
        HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
        [id(0x00000065), propget]
        HRESULT _stdcall Dependents([out, retval]
ICodeWarriorTargetFileCollection** pval);
        [id(0x00000066), propget]
        HRESULT _stdcall Dependencies([out, retval]
ICodeWarriorTargetFileCollection** pval);
        [id(0x00000067), propput]
        HRESULT _stdcall DebugInfo([in] VARIANT_BOOL value);
        [id(0x00000067), propget]
        HRESULT _stdcall DebugInfo([out, retval] VARIANT_BOOL*
value);
        [id(0x00000068), propput]
        HRESULT _stdcall InitBefore([in] VARIANT_BOOL value);
        [id(0x00000068), propget]
        HRESULT _stdcall InitBefore([out, retval] VARIANT_BOOL*
value);
        [id(0x00000069), propput]
        HRESULT _stdcall MergeLibrary([in] VARIANT_BOOL value);
        [id(0x00000069), propget]
        HRESULT _stdcall MergeLibrary([out, retval] VARIANT_BOOL*
value);
        [id(0x0000006a), propput]

```

```

        HRESULT _stdcall WeakImport([in] VARIANT_BOOL value);
        [id(0x0000006a), propget]
        HRESULT _stdcall WeakImport([out, retval] VARIANT_BOOL*
value);
    };

    [
        odl,
        uuid(BACE41C0-6DE6-11D2-AD83-006008A5C0A5),
        dual,
        oleautomation
    ]
    interface ICodeWarriorAccessPaths : IDispatch {
        [id(0x00000064), propget]
        HRESULT _stdcall UserAccessPaths([out, retval]
ICodeWarriorAccessPathCollection** pval);
        [id(0x00000065), propget]
        HRESULT _stdcall SystemAccessPaths([out, retval]
ICodeWarriorAccessPathCollection** pval);
        [id(0x00000066), propget]
        HRESULT _stdcall AlwaysSearchUserPaths([out, retval]
VARIANT_BOOL* pval);
        [id(0x00000066), propput]
        HRESULT _stdcall AlwaysSearchUserPaths([in] VARIANT_BOOL
pval);
        [id(0x00000068)]
        HRESULT _stdcall CreateAccessPath(
            [in] BSTR path,
            [in] VARIANT_BOOL Recursion,
            [in] EAccessPathLocation inLocation,
            [in] EAccessPathType inType,
            [out, retval] ICodeWarriorAccessPath**
pval);
        [id(0x0000006a)]
        HRESULT _stdcall CreateAccessPathByFileSpec(
            [in] IFileSpec* path,
            [in] VARIANT_BOOL Recursion,
            [in] EAccessPathLocation inLocation,
            [in] EAccessPathType inType,
            [out, retval] ICodeWarriorAccessPath**
pval);
        [id(0x00000069)]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        HRESULT _stdcall ApplyChanges();
};

[
    odl,
    uuid(916DA200-6DE6-11D2-AD83-006008A5C0A5),
    dual,
    oleautomation
]
interface ICodeWarriorAccessPathCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorAccessPath**
pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorAccessPath* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorAccessPath*
pval);
};

[
    odl,
    uuid(833D6550-6DE6-11D2-AD83-006008A5C0A5),
    dual,
    oleautomation
]
interface ICodeWarriorAccessPath : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall Recursive([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000064), propput]
    HRESULT _stdcall Recursive([in] VARIANT_BOOL pval);
    [id(0x00000065), propget]
    HRESULT _stdcall path([out, retval] IFileSpec** pval);
    [id(0x00000066), propget]

```

```

        HRESULT _stdcall AccessPathLocation([out, retval]
EAccessPathLocation* pval);
        [id(0x00000066), propput]
        HRESULT _stdcall AccessPathLocation([in]
EAccessPathLocation pval);
        [id(0x00000067), propget]
        HRESULT _stdcall AccessPathType([out, retval]
EAccessPathType* pval);
        [id(0x00000068), propget]
        HRESULT _stdcall UserTree([out, retval]
ICodeWarriorUserTree** pval);
        [id(0x00000069), propget]
        HRESULT _stdcall SubDirectories([out, retval]
ICodeWarriorAccessPathCollection** pval);
    };

    typedef enum {
        kAbsolute = 0,
        kProjectRelative = 1,
        kCompilerRelative = 2,
        kSystemRelative = 3,
        kUserDefined = 4
    } EAccessPathLocation;

    typedef enum {
        kUserPath = 0,
        kSystemPath = 1
    } EAccessPathType;

    [
        odl,
        uuid(50993290-6DE6-11D2-AD83-006008A5C0A5),
        dual,
        oleautomation
    ]
    interface ICodeWarriorUserTree : IDispatch {
        [id(0x00000001), propget]
        HRESULT _stdcall Name([out, retval] BSTR* pval);
        [id(0x00000001), propput]
        HRESULT _stdcall Name([in] BSTR pval);
        [id(0x00000064), propget]
        HRESULT _stdcall value([out, retval] BSTR* pval);
    };

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
        [id(0x00000064), propput]
        HRESULT _stdcall value([in] BSTR pval);
        [id(0x00000065), propget]
        HRESULT _stdcall type([out, retval] EUserDefinedTree*
val);
        [id(0x00000065), propput]
        HRESULT _stdcall type([in] EUserDefinedTree val);
};

typedef enum {
    kAbsolutePath = 0,
    kEnvironment = 1,
    kRegistry = 2
} EUserDefinedTree;

[
    odl,
    uuid(A7B77820-6DE6-11D2-AD83-006008A5C0A5),
    dual,
    oleautomation
]
interface ICodeWarriorUserTreeCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorUserTree** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorUserTree* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorUserTree* pval);
};

[
    odl,
    uuid(8463C22C-7E72-11D2-9A8E-00C04F79DE48),
```



```

    dual,
    oleautomation
]
interface ICodeWarriorSubTargetCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
pval);
        [in] long index,
        [out, retval] ICodeWarriorSubTarget**
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorSubTarget* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorSubTarget* pval);
};

[
    odl,
    uuid(CF40CE56-95FC-11D2-9A8D-00C04F79DE48),
    dual,
    oleautomation
]
interface ICodeWarriorSubTarget : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall Target([out, retval] ICodeWarriorTarget**
pval);
    [id(0x00000065), propget]
    HRESULT _stdcall LinkAgainstOutput([out, retval]
VARIANT_BOOL* pval);
};

[
    odl,
    uuid(F49299BE-C072-11D2-9ADC-00C04F79DE48),
    dual,
    oleautomation

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

]
interface IFileSpecCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] IFileSpec** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] IFileSpec* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] IFileSpec* pval);
};

[
    odl,
    uuid(E1179B70-EB6F-11D2-ADDA-00C04F804195),
    dual,
    oleautomation
]
interface IBSTRCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] BSTR* pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] BSTR val);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] BSTR val);
};

```

```

typedef enum {
    kStoreInProjectFile = 0,
    kStoreInTargetDataFile = 1,
    kStoreInProjectSettingsFile = 2
} EPluginDataStorageLoc;

[
    odl,
    uuid(0000000C-0000-0000-C000-000000000046)
]
interface IStream : ISequentialStream {
    HRESULT _stdcall RemoteSeek(
        [in] _LARGE_INTEGER dlibMove,
        [in] unsigned long dwOrigin,
        [out] _ULARGE_INTEGER* plibNewPosition);
    HRESULT _stdcall SetSize([in] _ULARGE_INTEGER libNewSize);
    HRESULT _stdcall RemoteCopyTo(
        [in] IStream* pstm,
        [in] _ULARGE_INTEGER cb,
        [out] _ULARGE_INTEGER* pcbRead,
        [out] _ULARGE_INTEGER* pcbWritten);
    HRESULT _stdcall Commit([in] unsigned long
grfCommitFlags);
    HRESULT _stdcall Revert();
    HRESULT _stdcall LockRegion(
        [in] _ULARGE_INTEGER libOffset,
        [in] _ULARGE_INTEGER cb,
        [in] unsigned long dwLockType);
    HRESULT _stdcall UnlockRegion(
        [in] _ULARGE_INTEGER libOffset,
        [in] _ULARGE_INTEGER cb,
        [in] unsigned long dwLockType);
    HRESULT _stdcall Stat(
        [out] tagSTATSTG* pstatstg,
        [in] unsigned long grfStatFlag);
    HRESULT _stdcall Clone([out] IStream** ppstm);
};

[
    odl,
    uuid(0C733A30-2A1C-11CE-ADE5-00AA0044773D)

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

]
interface ISequentialStream : IUnknown {
    HRESULT _stdcall RemoteRead(
        [out] char* pv,
        [in] unsigned long cb,
        [out] unsigned long* pcbRead);
    HRESULT _stdcall RemoteWrite(
        [in] char* pv,
        [in] unsigned long cb,
        [out] unsigned long* pcbWritten);
};

typedef struct tag_LARGE_INTEGER {
int64 QuadPart;
} _LARGE_INTEGER;

typedef struct tag_ULARGE_INTEGER {
uint64 QuadPart;
} _ULARGE_INTEGER;

typedef struct tagtagSTATSTG {
LPWSTR pwcsName;

unsigned long type;

_ULARGE_INTEGER cbSize;

_FILETIME mtime;

_FILETIME ctime;

_FILETIME atime;

unsigned long grfMode;

unsigned long grfLocksSupported;

GUID clsid;

```

```

unsigned long grfStateBits;

unsigned long reserved;
    } tagSTATSTG;

    typedef struct tag_FILETIME {

unsigned long dwLowDateTime;

unsigned long dwHighDateTime;
    } _FILETIME;

    [
        odl,
        uuid(6980FC87-A00A-11D2-9AB2-00C04F79DE48)
    ]
    interface ICodeWarriorBuildMessages : IDispatch {
        [propget]
            HRESULT _stdcall Errors([out]
ICodeWarriorMessageCollection** Errors);
        [propget]
            HRESULT _stdcall Warnings([out]
ICodeWarriorMessageCollection** Warnings);
        [propget]
            HRESULT _stdcall Informations([out]
ICodeWarriorMessageCollection** info);
        [propget]
            HRESULT _stdcall Definitions([out]
ICodeWarriorMessageCollection** Errors);
        [propget]
            HRESULT _stdcall ErrorCount([out] long* Count);
        [propget]
            HRESULT _stdcall WarningCount([out] long* Count);
        [propget]
            HRESULT _stdcall InformationCount([out] long* Count);
        [propget]
            HRESULT _stdcall DefinitionCount([out] long* Count);
    };

    [
        odl,
        uuid(A341D251-A00B-11D2-9AB2-00C04F79DE48),

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
    dual,
    oleautomation
]
interface ICodeWarriorMessageCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorMessage** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorMessage* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorMessage* pval);
};

[
    odl,
    uuid(DF1E763E-96A6-11D2-9AA9-00C04F79DE48)
]
interface ICodeWarriorMessage : IDispatch {
    [propget]
    HRESULT _stdcall type([out] EMsgType* type);
    [propget]
    HRESULT _stdcall FileSpec([out] IFileSpec** FileSpec);
    [propget]
    HRESULT _stdcall projectFile([out]
ICodeWarriorProjectFile** projectFile);
    [propget]
    HRESULT _stdcall MessageText([out] BSTR* message);
    [propget]
    HRESULT _stdcall ErrorNumber([out] long* ErrorNumber);
    [propget]
    HRESULT _stdcall Target([out] ICodeWarriorTarget**
Target);
    [propget]
    HRESULT _stdcall SourceOffset([out] long* offset);
```

```

    [propget]
    HRESULT _stdcall SourceLength([out] long* length);
    [propget]
    HRESULT _stdcall SourceLineNumber([out] long* lineNumber);
    [propget]
    HRESULT _stdcall TokenOffset([out] long* TokenOffset);
    [propget]
    HRESULT _stdcall TokenLength([out] long* TokenLength);
    [propget]
    HRESULT _stdcall MessageLineCount([out] long* lineCount);
    [propget]
    HRESULT _stdcall MessageLength([out] long* MessageLength);
};

typedef enum {
    typeNotDefined = 0,
    typeInformation = 1,
    typeWarning = 2,
    typeError = 3,
    typeDefinition = 4
} EMsgType;

[
    odl,
    uuid(6971AB76-EC83-11D2-9B0C-00C04F79DE48),
    dual,
    oleautomation
]
interface ICodeWarriorTargetOutput : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall OutputKind([out, retval]
ECodeWarriorTargetOutputKind* kind);
    [id(0x00000009), propget]
    HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
};

typedef enum {
    kCWOutputNone = 0,
    kCWOutputFile = 1,
    kCWOutputDirectory = 2
} ECodeWarriorTargetOutputKind;

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

    [
        odl,
        uuid(5EC306A1-283D-11D0-989C-0080C74ADF8C),
        dual,
        oleautomation
    ]
interface ICodeWarriorApp : IDispatch {
    [id(0x00000066), propget]
    HRESULT _stdcall Application([out, retval] IDispatch**
pval);
    [id(0x00000067), propget]
    HRESULT _stdcall FullName([out, retval] BSTR* pval);
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000004), propget]
    HRESULT _stdcall Visible([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000004), propput]
    HRESULT _stdcall Visible([in] VARIANT_BOOL pval);
    [id(0x00000069), propget]
    HRESULT _stdcall Projects([out, retval]
ICodeWarriorProjectCollection** pval);
    [id(0x0000006c), propget]
    HRESULT _stdcall CreatableItems([out, retval]
ICodeWarriorCreatableItemCollection** pval);
    [id(0x00000077), propget]
    HRESULT _stdcall Documents([out, retval]
ICodeWarriorDocumentCollection** pval);
    [id(0x00000078), propget]
    HRESULT _stdcall ActiveDocument([out, retval]
ICodeWarriorDocument** pval);
    [id(0x00000083), propget]
    HRESULT _stdcall DefaultProjectDocument([out, retval]
ICodeWarriorProjectDocument** pval);
    [id(0x0000007b), propget]
    HRESULT _stdcall DefaultProject([out, retval]
ICodeWarriorProject** Project);
    [id(0x0000007c), propput]
    HRESULT _stdcall AllowUserInteraction([in] VARIANT_BOOL
rhs);
    [id(0x0000007d), propget]

```



```

        HRESULT _stdcall VersionControl([out, retval]
ICodeWarriorVersionControl** vcs);
        [id(0x0000006d)]
        HRESULT _stdcall CreateProject(
                [in] BSTR filePath,
                [in] BSTR linkerName,
                [in] BSTR designName,
                [in] BSTR targetName,
                [in] VARIANT_BOOL fMakeVisible,
                [out, retval] ICodeWarriorProject** pval);
        [id(0x0000006e)]
        HRESULT _stdcall CreateProjectByFileSpec(
                [in] IFileSpec* projectFileSpec,
                [in] BSTR linkerName,
                [in] BSTR designName,
                [in] BSTR targetName,
                [in] IFileSpec* stationeryFileSpec,
                [in] VARIANT_BOOL fMakeVisible,
                [out, retval] ICodeWarriorProject** pval);
        [id(0x00000080)]
        HRESULT _stdcall ImportProject(
                [in] BSTR textFilePath,
                [in] BSTR projectFilePath,
                [in] VARIANT_BOOL fMakeVisible,
                [out, retval] ICodeWarriorProject** pval);
        [id(0x00000081)]
        HRESULT _stdcall ImportProjectByFileSpec(
                [in] IFileSpec* textFileSpec,
                [in] IFileSpec* projectFileSpec,
                [in] VARIANT_BOOL fMakeVisible,
                [out, retval] ICodeWarriorProject** pval);
        [id(0x00000065)]
        HRESULT _stdcall OpenProject(
                [in] BSTR filePath,
                [in] VARIANT_BOOL fMakeVisible,
                [in] ECodeWarriorConvertOption
convertOption,
                [in] ECodeWarriorRevertPanelOption
revertOption,
                [out, retval] ICodeWarriorProject** pval);
        [id(0x0000006f)]
        HRESULT _stdcall OpenProjectByFileSpec(

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

[in] IFileSpec* FileSpec,
[in] VARIANT_BOOL fMakeVisible,
[in] ECodeWarriorConvertOption
convertOption,
[in] ECodeWarriorRevertPanelOption
revertOption,
[out, retval] ICodeWarriorProject** pval);
[id(0x0000006a)]
HRESULT _stdcall AddCreatableItem([in] IUnknown* Item);
[id(0x0000006b)]
HRESULT _stdcall RemoveCreatableItem([in] IUnknown* Item);
[id(0x00000070)]
HRESULT _stdcall FindLogicalFolder(
[in] BSTR folderName,
[out, retval] IFileSpec** folder);
[id(0x00000071)]
HRESULT _stdcall FindDesignForDataModel(
[in] IUnknown* DataModel,
[out, retval] ICodeWarriorDesign**
Project);
[id(0x00000072)]
HRESULT _stdcall GetNamedPluginData(
[in] BSTR resourceName,
[out] IStream** pluginData);
[id(0x00000073)]
HRESULT _stdcall SetNamedPluginData(
[in] BSTR resourceName,
[in] IStream* pluginData);
[id(0x00000074)]
HRESULT _stdcall RemoveNamedPluginData([in] BSTR
resourceName);
[id(0x00000075)]
HRESULT _stdcall GetSetting(
[in] BSTR settingsName,
[out, retval] VARIANT* pval);
[id(0x00000076)]
HRESULT _stdcall SetSetting(
[in] BSTR settingsName,
[in] VARIANT pval);
[id(0x00000079)]
HRESULT _stdcall OpenTextDocumentByFileSpec(
[in] IFileSpec* FileSpec,

```

```

        [in] VARIANT_BOOL create,
        [out, retval] ICodeWarriorTextDocument**
document);
    [id(0x0000007a)]
    HRESULT _stdcall OpenTextDocument(
        [in] BSTR inPath,
        [in] VARIANT_BOOL create,
        [out, retval] ICodeWarriorTextDocument**
document);
    [id(0x00000084)]
    HRESULT _stdcall OpenUntitledTextDocument([out, retval]
ICodeWarriorTextDocument** document);
    [id(0x0000007e)]
    HRESULT _stdcall AttemptModify(
        [in] IFileSpec* FileSpec,
        [in] ECodeWarriorVCSInteractionOption
uiParameter,
        [in] ICodeWarriorProject* Project);
    [id(0x0000007f)]
    HRESULT _stdcall DoCommand([in] long commandID);
    [id(0x00000082)]
    HRESULT _stdcall QueueDeferredAction(IUnknown* action);
    [id(0x00000085)]
    HRESULT _stdcall IsBuildInProgress([out, retval]
VARIANT_BOOL* pval);
};

[
    odl,
    uuid(1A657F50-95B5-11D1-B31B-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorProjectCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(0x00000006), propget]
    HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
    [id(00000000)]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorProject** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorProject* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorProject* pval);
};

[
    odl,
    uuid(6161C790-FB3B-11D1-B331-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorCreatableItemCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval] ICodeWarriorCreatableItem**
pval);
};

[
    odl,
    uuid(145691E0-FA29-11D1-B330-0060081C5489)
]
interface ICodeWarriorCreatableItem : IUnknown {
    HRESULT _stdcall GetDisplayName([out] BSTR* __MIDL_0017);
    HRESULT _stdcall GetIcon(
        IUnknown* iconList,
        int* index);
    HRESULT _stdcall GetCategory([out] BSTR* __MIDL_0018);
    HRESULT _stdcall InvokesWizard();
};

[
    odl,

```

```

        uuid(BA875690-B46A-11D2-ADB6-00C04F804195),
        dual,
        oleautomation
    ]
    interface ICodeWarriorDocumentCollection : IDispatch {
        [id(0x00000002), propget]
        HRESULT _stdcall Count([out, retval] long* pval);
        [id(0x00000006), propget]
        HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
        [id(0x00000003), propget]
        HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
        [id(00000000)]
        HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorDocument** pval);
        [id(0x00000007)]
        HRESULT _stdcall Add([in] ICodeWarriorDocument* pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in] ICodeWarriorDocument* pval);
    };

    [
        odl,
        uuid(08A1D280-B468-11D2-ADB6-00C04F804195),
        dual,
        oleautomation
    ]
    interface ICodeWarriorDocument : IDispatch {
        [id(0x00000001), propget]
        HRESULT _stdcall Name([out, retval] BSTR* pval);
        [id(0x00000009), propget]
        HRESULT _stdcall FileSpec([out, retval] IFileSpec** pval);
        [id(0x00000064), propget]
        HRESULT _stdcall ActiveDocument([out, retval]
VARIANT_BOOL* pval);
        [id(0x00000006), propget]
        HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL*
pval);
        [id(0x00000065), propget]
        HRESULT _stdcall Dirty([out, retval] VARIANT_BOOL* pval);
        [id(0x00000004), propget]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

    HRESULT _stdcall Visible([out, retval] VARIANT_BOOL*
pval);
    [id(0x00000004), propput]
    HRESULT _stdcall Visible([in] VARIANT_BOOL pval);
    [id(0x00000069), propget]
    HRESULT _stdcall XPos([out, retval] int* pval);
    [id(0x00000069), propput]
    HRESULT _stdcall XPos([in] int pval);
    [id(0x0000006a), propget]
    HRESULT _stdcall YPos([out, retval] int* pval);
    [id(0x0000006a), propput]
    HRESULT _stdcall YPos([in] int pval);
    [id(0x0000006b), propget]
    HRESULT _stdcall Width([out, retval] int* pval);
    [id(0x0000006b), propput]
    HRESULT _stdcall Width([in] int pval);
    [id(0x0000006c), propget]
    HRESULT _stdcall Height([out, retval] int* pval);
    [id(0x0000006c), propput]
    HRESULT _stdcall Height([in] int pval);
    [id(0x00000066)]
    HRESULT _stdcall Save();
    [id(0x00000067)]
    HRESULT _stdcall Close([in] VARIANT_BOOL bSaveChanges);
    [id(0x00000068)]
    HRESULT _stdcall Activate();
};

[
    odl,
    uuid(41A67F00-B584-11D2-ADB6-00C04F804195),
    dual,
    oleautomation
]
interface ICodeWarriorProjectDocument : ICodeWarriorDocument
{
    [id(0x000000c8), propget]
    HRESULT _stdcall Project([out, retval]
ICodeWarriorProject** pval);
    [id(0x000000c9)]
    HRESULT _stdcall SelectFiles(

```

```

        [in] ICodeWarriorProjectFileCollection*
projectFiles,
        [in] VARIANT_BOOL select);
    [id(0x000000ca)]
    HRESULT _stdcall SelectedFiles([out, retval]
ICodeWarriorProjectFileCollection** projectFiles);
    [id(0x000000cb)]
    HRESULT _stdcall ExpandGroup([in] BSTR groupName);
    [id(0x000000cc)]
    HRESULT _stdcall CollapseGroup([in] BSTR groupName);
};

[
    odl,
    uuid(5C5A784E-C070-11D2-9ADC-00C04F79DE48)
]
interface ICodeWarriorVersionControl : IDispatch {
    [propget]
    HRESULT _stdcall Name([out, retval] BSTR* vcsName);
    HRESULT _stdcall GetVCSState(
        [in] IFileSpec* FileSpec,
        [out, retval] ICodeWarriorVCSState**
VCSState);
    HRESULT _stdcall CheckIn([in] IFileSpecCollection*
fileSpecCollection);
    HRESULT _stdcall CheckOut([in] IFileSpecCollection*
fileSpecCollection);
    HRESULT _stdcall UnLock([in] IFileSpecCollection*
fileSpecCollection);
    HRESULT _stdcall Get([in] IFileSpecCollection*
fileSpecCollection);
    HRESULT _stdcall UndoCheckOut([in] IFileSpecCollection*
fileSpecCollection);
    HRESULT _stdcall Connect();
    HRESULT _stdcall Disconnect();
    HRESULT _stdcall IsConnected([out, retval] VARIANT_BOOL*
pval);
};

typedef enum {
    kCWConvertYes = 0,
    kCWConvertNo = 1,

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        kCWConvertAsk = 2
    } ECodeWarriorConvertOption;

typedef enum {
    kCWDoNotRevertPanel = 0,
    kCWAllowPanelRevert = 1
} ECodeWarriorRevertPanelOption;

[
    odl,
    uuid(1AD264D0-B46C-11D2-ADB6-00C04F804195),
    dual,
    oleautomation
]
interface ICodeWarriorTextDocument : ICodeWarriorDocument {
    [id(0x000000c8), propget]
    HRESULT _stdcall TextEngine([out, retval]
ICodeWarriorTextEngine** pval);
    [id(0x000000cb)]
    HRESULT _stdcall SaveAsByFileSpec([in] IFileSpec*
FileSpec);
    [id(0x000000c9)]
    HRESULT _stdcall SaveAs([in] BSTR val);
    [id(0x000000cc)]
    HRESULT _stdcall SaveACopyAsByFileSpec([in] IFileSpec*
FileSpec);
    [id(0x000000ca)]
    HRESULT _stdcall SaveACopyAs([in] BSTR val);
};

[
    odl,
    uuid(B31823F0-B470-11D2-ADB6-00C04F804195),
    dual,
    oleautomation
]
interface ICodeWarriorTextEngine : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall SelectionStart([out, retval] int* pval);
    [id(0x00000064), propput]
    HRESULT _stdcall SelectionStart([in] int pval);
    [id(0x00000065), propget]

```



```

HRESULT _stdcall SelectionEnd([out, retval] int* pval);
[id(0x00000065), propput]
HRESULT _stdcall SelectionEnd([in] int pval);
[id(0x00000066), propget]
HRESULT _stdcall SelectionLineStart([out, retval] int*
pval);
[id(0x00000066), propput]
HRESULT _stdcall SelectionLineStart([in] int pval);
[id(0x00000067), propget]
HRESULT _stdcall SelectionLineEnd([out, retval] int*
pval);
[id(0x00000067), propput]
HRESULT _stdcall SelectionLineEnd([in] int pval);
[id(0x00000068), propget]
HRESULT _stdcall HasSelection([out, retval] VARIANT_BOOL*
pval);
[id(0x00000069), propget]
HRESULT _stdcall SelectionText([out, retval] BSTR* pval);
[id(0x00000069), propput]
HRESULT _stdcall SelectionText([in] BSTR pval);
[id(0x0000006a), propget]
HRESULT _stdcall lineCount([out, retval] int* pval);
[id(0x0000006b), propget]
HRESULT _stdcall TextLength([out, retval] int* pval);
[id(0x0000006c)]
HRESULT _stdcall GetTextForOffsetRange(
[in] int selStart,
[in] int selEnd,
[out, retval] BSTR* pval);
[id(0x0000006d)]
HRESULT _stdcall GetTextForLineRange(
[in] int lineStart,
[in] int lineEnd,
[out, retval] BSTR* pval);
[id(0x0000006e)]
HRESULT _stdcall GetLineForOffset(
[in] int offset,
[out, retval] int* line);
[id(0x0000006f)]
HRESULT _stdcall GetOffsetForLine(
[in] int line,
[out, retval] int* offset);

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
    [id(0x00000070)]
    HRESULT _stdcall InsertText([in] BSTR val);
};

typedef enum {
    kCWAsk = 0,
    kCWDoNothing = 1,
    kCWUseDefault = 2
} ECodeWarriorVCSInteractionOption;

typedef enum {
    kReportMsgAlert = 0,
    kReportMsgInformation = 1,
    kReportMsgWarning = 2
} EReportMsgType;

[
    odl,
    uuid(AE200FB0-5C69-11D2-8120-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorComponent : IDispatch {
    [id(0x00000064), propget]
    HRESULT _stdcall Class([out, retval] ICodeWarriorClass**
pval);
    [id(0x00000065), propget]
    HRESULT _stdcall Properties([out, retval]
ICodeWarriorComponentPropertyCollection** pval);
    [id(0x00000066), propget]
    HRESULT _stdcall Methods([out, retval]
ICodeWarriorMethodCollection** pval);
    [id(0x00000067), propget]
    HRESULT _stdcall EventSets([out, retval]
ICodeWarriorComponentEventSetCollection** pval);
    [id(0x00000068), propget]
    HRESULT _stdcall CanHaveMultipleEventSets([out, retval]
long* pval);
    [id(0x00000069), propget]
    HRESULT _stdcall DefaultEvent([out, retval]
ICodeWarriorComponentEvent** pval);
    [id(0x0000006a), propget]
```

```

        HRESULT _stdcall EventConnectionsEnabled([out, retval]
long* pval);
    };

    [
        odl,
        uuid(AE200FB5-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
interface ICodeWarriorComponentPropertyCollection : IDispatch
{
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
        [in] long index,
        [out, retval]
ICodeWarriorComponentProperty** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorComponentProperty*
pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in]
ICodeWarriorComponentProperty* pval);
};

    [
        odl,
        uuid(AE200FB1-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
interface ICodeWarriorComponentProperty : IDispatch {
    [id(0x00000001), propget]
    HRESULT _stdcall Name([out, retval] BSTR* pval);
    [id(0x00000064), propget]
    HRESULT _stdcall type([out, retval] BSTR* pval);
    [id(0x00000065), propget]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        HRESULT _stdcall Getter([out, retval] ICodeWarriorMethod**
pval);
        [id(0x00000066), propget]
        HRESULT _stdcall Setter([out, retval] ICodeWarriorMethod**
pval);
    };

    [
        odl,
        uuid(AE200FB6-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
    interface ICodeWarriorComponentEventSetCollection : IDispatch
    {
        [id(0x00000002), propget]
        HRESULT _stdcall Count([out, retval] long* pval);
        [id(0x00000003), propget]
        HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
        [id(00000000)]
        HRESULT _stdcall Item(
            [in] long index,
            [out, retval]
ICodeWarriorComponentEventSet** pval);
        [id(0x00000007)]
        HRESULT _stdcall Add([in] ICodeWarriorComponentEventSet*
pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in]
ICodeWarriorComponentEventSet* pval);
    };

    [
        odl,
        uuid(AE200FB2-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
    interface ICodeWarriorComponentEventSet : IDispatch {
        [id(0x00000064), propget]
        HRESULT _stdcall Class([out, retval] ICodeWarriorClass**
pval);

```

```

        [id(0x00000065), propget]
        HRESULT _stdcall EventSetName([out, retval] BSTR* pval);
        [id(0x00000066), propget]
        HRESULT _stdcall Events([out, retval]
ICodeWarriorComponentEventCollection** pval);
    };

    [
        odl,
        uuid(AE200FB7-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
    interface ICodeWarriorComponentEventCollection : IDispatch {
        [id(0x00000002), propget]
        HRESULT _stdcall Count([out, retval] long* pval);
        [id(0x00000003), propget]
        HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
        [id(00000000)]
        HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorComponentEvent**
pval);
        [id(0x00000007)]
        HRESULT _stdcall Add([in] ICodeWarriorComponentEvent*
pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in] ICodeWarriorComponentEvent*
pval);
    };

    [
        odl,
        uuid(AE200FB3-5C69-11D2-8120-006008C3EEF1),
        dual,
        oleautomation
    ]
    interface ICodeWarriorComponentEvent : IDispatch {
        [id(0x00000001), propget]
        HRESULT _stdcall Name([out, retval] BSTR* pval);
        [id(0x00000064), propget]

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

        HRESULT _stdcall Method([out, retval] ICodeWarriorMethod**
pval);
        [id(0x00000065), propget]
        HRESULT _stdcall EventSet([out, retval]
ICodeWarriorComponentEventSet** pval);
        [id(0x00000066)]
        HRESULT _stdcall GetDefaultMethodName(
                [in] IUnknown* modelobject,
                [out] BSTR* pdefname);
};

[
    odl,
    uuid(2F48B6D5-052A-11D2-80C5-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorSymbolCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]
    HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
    [id(00000000)]
    HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorSymbol** pval);
    [id(0x00000007)]
    HRESULT _stdcall Add([in] ICodeWarriorSymbol* pval);
    [id(0x00000008)]
    HRESULT _stdcall Remove([in] ICodeWarriorSymbol* pval);
};

[
    odl,
    uuid(AE200FB4-5C69-11D2-8120-006008C3EEF1),
    dual,
    oleautomation
]
interface ICodeWarriorComponentCollection : IDispatch {
    [id(0x00000002), propget]
    HRESULT _stdcall Count([out, retval] long* pval);
    [id(0x00000003), propget]

```

```

        HRESULT _stdcall _NewEnum([out, retval] IDispatch** pval);
        [id(00000000)]
        HRESULT _stdcall Item(
            [in] long index,
            [out, retval] ICodeWarriorComponent**
pval);
        [id(0x00000007)]
        HRESULT _stdcall Add([in] ICodeWarriorComponent* pval);
        [id(0x00000008)]
        HRESULT _stdcall Remove([in] ICodeWarriorComponent* pval);
    };

typedef enum {
    kMaximumTargetNameLength = 31
} ECodeWarriorProjectConstants;

[
    uuid(D6D02BB0-ACCC-11D2-ADB3-00C04F804195),
    appobject
]
coclass CodeWarriorApp {
    [default] interface ICodeWarriorApp;
    [default, source] interface ICodeWarriorAppEvents;
};

[
    odl,
    uuid(5EC306A3-283D-11D0-989C-0080C74ADF8C)
]
interface ICodeWarriorAppEvents : IUnknown {
    HRESULT _stdcall Startup();
    HRESULT _stdcall QueryQuit();
    HRESULT _stdcall Quit();
    HRESULT _stdcall ProjectOpened(
        [in] ICodeWarriorProject* Project,
        VARIANT_BOOL fVisible);
    HRESULT _stdcall ProjectVisible([in] ICodeWarriorProject*
Project);
    HRESULT _stdcall DataModelCreated(
        [in] IUnknown* DataModel,
        VARIANT_BOOL fFromStorage);
};

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
        HRESULT _stdcall DataModelLoaded([in] IUnknown*
DataModel);
    };

    typedef enum {
        kCWChoiceCheckSyntax = 0,
        kCWChoicePreprocess = 1,
        kCWChoicePrecompile = 2,
        kCWChoiceCompile = 3,
        kCWChoiceDisassemble = 4
    } ECodeWarriorCompileChoice;

    [
        uuid(7153E430-AE65-11D2-ADB4-00C04F804195)
    ]
    coclass CWAutomationProject {
        [default] interface ICodeWarriorProject;
        [default, source] interface ICodeWarriorProjectEvents;
    };

    [
        odl,
        uuid(3D3B7F80-9694-11D1-B31B-0060081C5489)
    ]
    interface ICodeWarriorProjectEvents : IUnknown {
        HRESULT _stdcall QueryUIClose([in] ICodeWarriorProject*
Project);
        HRESULT _stdcall VisibleChanged(
            [in] ICodeWarriorProject* Project,
            [in] VARIANT_BOOL fVisible);
        HRESULT _stdcall ProjectClosing([in] ICodeWarriorProject*
Project);
        HRESULT _stdcall DesignCreated([in] ICodeWarriorDesign*
Design);
        HRESULT _stdcall QueryDeleteDesign([in]
ICodeWarriorDesign* Design);
        HRESULT _stdcall DeletingDesign([in] ICodeWarriorDesign*
Design);
        HRESULT _stdcall BuildStarted(
            [in] ECodeWarriorCompileChoice choice,
            [in] long buildID,
```



```

                                [in] ICodeWarriorTargetCollection*
targetList);
    HRESULT _stdcall BuildEnded(
        [in] ECodeWarriorCompileChoice choice,
        [in] long buildID,
        [in] VARIANT_BOOL fBuildSucceeded,
        [out] ICodeWarriorBuildMessages*
buildMessages);
    HRESULT _stdcall QueryAboutToBuild(
        [in] ECodeWarriorCompileChoice choice,
        [in] long buildID,
        [in] ICodeWarriorTargetCollection*
targetList);
    HRESULT _stdcall RevertCompleted();
};

[
    uuid(92794C60-AE65-11D2-ADB4-00C04F804195)
]
coclass CWAutomationTarget {
    [default] interface ICodeWarriorTarget;
};

[
    uuid(A4C352E0-AE65-11D2-ADB4-00C04F804195)
]
coclass CWAutomationDesign {
    [default] interface ICodeWarriorDesign;
    [default, source] interface ICodeWarriorDesignEvents;
};

[
    odl,
    uuid(51FB0BD0-E515-11D1-B32A-0060081C5489)
]
interface ICodeWarriorDesignEvents : IUnknown {
    HRESULT _stdcall TargetAdded([in] ICodeWarriorTarget*
Target);
    HRESULT _stdcall RemovingTarget([in] ICodeWarriorTarget*
Target);
};

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
[
    odl,
    uuid(8967DC00-57CD-11D2-B358-0060081C5489)
]
interface ICodeWarriorDesignAttachment : IUnknown {
    HRESULT _stdcall DesignInitialized(ICodeWarriorDesign*
__MIDL_0014);
    HRESULT _stdcall DesignClosing(ICodeWarriorDesign*
__MIDL_0015);
    HRESULT _stdcall RemovingAttachment(ICodeWarriorDesign*
__MIDL_0016);
};

[
    uuid(BB058510-AE65-11D2-ADB4-00C04F804195)
]
coclass CWAutomationAccessPath {
    [default] interface ICodeWarriorAccessPath;
};

[
    uuid(D69AC280-AE65-11D2-ADB4-00C04F804195)
]
coclass CWAutomationAccessPaths {
    [default] interface ICodeWarriorAccessPaths;
};

[
    uuid(B4B07CB0-B467-11D2-ADB6-00C04F804195)
]
coclass CWAutomationDocument {
    [default] interface ICodeWarriorDocument;
};

[
    uuid(0ADC5170-B46C-11D2-ADB6-00C04F804195)
]
coclass CWAutomationTextDocument {
    [default] interface ICodeWarriorTextDocument;
};

[
```

```

    uuid(52247090-B583-11D2-ADB6-00C04F804195)
]
coclass CWAutomationProjectDocument {
    [default] interface ICodeWarriorProjectDocument;
};

[
    uuid(8F920920-B46C-11D2-ADB6-00C04F804195)
]
coclass CWAutomationTextEngine {
    [default] interface ICodeWarriorTextEngine;
};

typedef enum {
    newIconProject = -1,
    newIconTextFile = -2,
    newIconCatalog = -3
} __MIDL__MIDL_itf_CodeWarriorComIntf_0114_0001;

typedef [public]
    __MIDL__MIDL_itf_CodeWarriorComIntf_0115_0001
ECreateProjectType;

typedef enum {
    createsProjectOnly = 0,
    createsDesign = 1,
    createsTargets = 2
} __MIDL__MIDL_itf_CodeWarriorComIntf_0115_0001;

[
    odl,
    uuid(229924D0-FA29-11D1-B330-0060081C5489),
    dual,
    oleautomation
]
interface ICodeWarriorCreateProjectItem :
ICodeWarriorCreatableItem {
    [id(0x60020000)]
    HRESULT _stdcall GetCreatedProjectType([out, retval]
ECreateProjectType* pval);
    [id(0x60020001)]

```



## CodeWarrior IDE Interface Definition Language (IDL)

---

```
        HRESULT _stdcall RequiresFileExtension([out, retval]
VARIANT_BOOL* pval);
        [id(0x60020002)]
        HRESULT _stdcall CreateNewProject([in] IFileSpec*
newFileSpec);
        [id(0x60020003)]
        HRESULT _stdcall CreateInExistingProject(
                [in] BSTR newItemName,
                [in] ICodeWarriorProject* Project);
};

[
    odl,
    uuid(229924D1-FA29-11D1-B330-0060081C5489)
]
interface ICodeWarriorCreateFileItem :
ICodeWarriorCreatableItem {
    HRESULT _stdcall CanCreateUntitledFile();
    HRESULT _stdcall CanAddFileToProject();
    HRESULT _stdcall CreateUntitledFile();
    HRESULT _stdcall CreateAndAddFile(
        [in] IFileSpec* newFileSpec,
        [in] ICodeWarriorProject* Project,
        [in] ICodeWarriorTargetCollection* Targets,
        [out] VARIANT_BOOL* fFilesAdded);
};

[
    odl,
    uuid(229924D3-FA29-11D1-B330-0060081C5489)
]
interface ICodeWarriorCreateObjectItem :
ICodeWarriorCreatableItem {
    HRESULT _stdcall AreObjectsCreatedInDesign();
    HRESULT _stdcall CreateObjectInDesign(
        [in] BSTR newItemName,
        [in] ICodeWarriorProject* Project,
        [in] ICodeWarriorDesign* Design);
    HRESULT _stdcall CreateObjectInTargets(
        [in] BSTR newItemName,
        [in] ICodeWarriorProject* Project,
```

```

                                [in] ICodeWarriorTargetCollection*
Targets);
    HRESULT _stdcall NeedsObjectName();
};

[
    uuid(B980537C-C37E-11D2-9ADF-00C04F79DE48),
    appobject
]
coclass CWCodeWarriorVCS {
    [default] interface ICodeWarriorVersionControl;
    [default, source] interface
ICodeWarriorVCSFileStateListener;
};

[
    odl,
    uuid(B980537A-C37E-11D2-9ADF-00C04F79DE48)
]
interface ICodeWarriorVCSFileStateListener : IUnknown {
    HRESULT _stdcall StateChanged(
        [in] IFileSpec* FileSpec,
        [in] ICodeWarriorVCSState* VCSState);
};

[
    odl,
    uuid(BE65AD59-C4BC-11D2-8065-006008C3EEB0),
    dual,
    oleautomation
]
interface ICodeWarriorProjectAssociation : IUnknown {
    [id(0x00000064), propget]
    HRESULT _stdcall Project([out, retval]
ICodeWarriorProject** pval);
    [id(0x00000064), propput]
    HRESULT _stdcall Project([in] ICodeWarriorProject* pval);
};

[
    odl,
    uuid(9DDD415E-AD7C-11D2-B26C-00C04F72E4D1),

```

## CodeWarrior IDE Interface Definition Language (IDL)

---

```

    dual,
    oleautomation
]
interface ICodeWarriorErrorInfo : IUnknown {
    [id(0x00000064), propput]
    HRESULT _stdcall action([in] BSTR actionStr);
    [id(0x00000064), propget]
    HRESULT _stdcall action([out, retval] BSTR* actionStr);
    [id(0x00000065), propput]
    HRESULT _stdcall Reason([in] BSTR actionStr);
    [id(0x00000065), propget]
    HRESULT _stdcall Reason([out, retval] BSTR* actionStr);
    [id(0x00000066), propput]
    HRESULT _stdcall MWErr([in] long err);
    [id(0x00000066), propget]
    HRESULT _stdcall MWErr([out, retval] long* err);
    [id(0x00000067), propput]
    HRESULT _stdcall HRESULT([in] HRESULT err);
    [id(0x00000067), propget]
    HRESULT _stdcall HRESULT([out, retval] HRESULT* err);
    [id(0x00000068), propput]
    HRESULT _stdcall DWORDErr([in] unsigned long err);
    [id(0x00000068), propget]
    HRESULT _stdcall DWORDErr([out, retval] unsigned long*
err);
    [id(0x00000069), propput]
    HRESULT _stdcall MacOSErr([in] short err);
    [id(0x00000069), propget]
    HRESULT _stdcall MacOSErr([out, retval] short* err);
    [id(0x0000006a), propput]
    HRESULT _stdcall Source([in] BSTR rhs);
    [id(0x0000006b), propput]
    HRESULT _stdcall HelpContext([in] unsigned long rhs);
    [id(0x0000006c), propput]
    HRESULT _stdcall HelpFile([in] BSTR rhs);
};
};

```

# Index

---

## A

- Access Paths 13
  - API Reference 13
  - Data Types 30
  - Interface
    - ICodeWarriorAccessPath 14
    - ICodeWarriorAccessPaths 19
    - ICodeWarriorUserTree 26
- Activate
  - ICodeWarriorDocument 160
- ActivateEvent
  - ICodeWarriorWindowEvents 399
- Add
  - Collection 79
- AddAttachment
  - ICodeWarriorDesign 130
- AddComponentAttachment
  - ICodeWarriorSymbolContainer 278
- AddCreatableItem
  - ICodeWarriorApp 36
- AddFile
  - ICodeWarriorDesign 131
  - ICodeWarriorTarget 287
- AddFile2
  - ICodeWarriorDesign 131
  - ICodeWarriorTarget 287
- AddFile2ByFileSpec
  - ICodeWarriorDesign 132
  - ICodeWarriorTarget 288
- AddFile2ByFileSpecCollection
  - ICodeWarriorTarget 289
- AddFileByFileSpec
  - ICodeWarriorDesign 133
  - ICodeWarriorTarget 290
- AddFileByFileSpecCollection
  - ICodeWarriorTarget 291
- AddSubTarget
  - ICodeWarriorTarget 291
- AddUserTree
  - ICodeWarriorApp 36
  - ICodeWarriorTarget 292
- API Reference
  - Access Paths 13
  - Application 34

- AppendMenuCommand
  - ICodeWarriorMenu 192
- Application
  - API Reference 34
  - Data Types 72
- Application Object, Working with 33
- ApplyChanges
  - ICodeWarriorAccessPaths 19
- AreObjectsCreatedInDesign
  - ICodeWarriorCreateObjectItem 120
- AssociateWindowWithProject 388
- AttemptModify
  - ICodeWarriorApp 37

## B

- bug fixes 10
- Build
  - ICodeWarriorProject 216
  - ICodeWarriorTarget 292
- BuildAgainstSubProjectTarget
  - ICodeWarriorTarget 293
- BuildAndWaitToComplete
  - ICodeWarriorProject 218
  - ICodeWarriorTarget 293
- BuildAndWaitToCompleteWithOptions 218
  - ICodeWarriorTarget 294
- BuildEnded
  - ICodeWarriorProjectEvents 239
- BuildPopupMenuList
  - ICodeWarriorPopupMenuToolbarItem 347
- BuildStarted
  - ICodeWarriorProjectEvents 240
- BuildWithOptions
  - ICodeWarriorProject 217
  - ICodeWarriorTarget 294

## C

- CanAddFileToProject
  - ICodeWarriorCreateFileItem 117
- CanCreateUntitledFile
  - ICodeWarriorCreateFileItem 117
- CheckIn
  - ICodeWarriorProjectFile 245
  - ICodeWarriorVersionControl 370

- CheckOut
  - ICodeWarriorProjectFile 245
  - ICodeWarriorVersionControl 370
- CleanupPopupMenuList
  - ICodeWarriorPopupMenuToolBarItem 348
- Clone
  - IFileSpec 185
- CloneTarget
  - ICodeWarriorProject 219
- Close
  - ICodeWarriorDocument 161
  - ICodeWarriorProject 220
- CollapseGroup
  - ICodeWarriorProjectDocument 168
- Collection
  - Add 79
  - get\_\_NewEnum 80
  - get\_Count 80
  - get\_ReadOnly 81
  - Item 81
  - Remove 82
- Collections API
  - Overview 77
  - Reference 78
- Command Status Data Type 95
- Commands API
  - Overview 83
  - Reference 88
  - Using 84
- CompareFiles
  - ICodeWarriorCompare 69
- CompareFilesByFileSpec
  - ICodeWarriorCompare 70
- CompareFolders
  - ICodeWarriorCompare 71
- CompileFiles
  - ICodeWarriorDesign 134
  - ICodeWarriorTarget 295
- CompileFilesAndWaitToComplete
  - ICodeWarriorDesign 134
  - ICodeWarriorTarget 296
- CompileFilesWithChoice
  - ICodeWarriorProject 220
  - ICodeWarriorTarget 296
- Components API
  - Overview 99
  - Reference 99
- Connect
  - ICodeWarriorVersionControl 371
- ContainsTarget
  - ICodeWarriorDesign 135
- conventions 10
- Copy
  - IFileSpec 186
- Creatable Item Category Constants Data Type 127
- Creatable Items Data Types 127
- CreateAccessPath
  - ICodeWarriorAccessPaths 19
- CreateAccessPathByFileSpec
  - ICodeWarriorAccessPaths 20
- CreateAccessPathByPosition
  - ICodeWarriorAccessPaths 21
- CreateAndAddFile
  - ICodeWarriorCreateFileItem 118
- CreateDesign
  - ICodeWarriorProject 221
- CreateInExistingProject
  - ICodeWarriorCreateProjectItem 124
- CreateItemControl
  - ICodeWarriorCustomToolBarItem 344
- CreateNewCommandGroup
  - ICodeWarriorCommandRegistry 92
- CreateNewProject
  - ICodeWarriorCreateProjectItem 125
- CreateObjectInDesign
  - ICodeWarriorCreateObjectItem 121
- CreateObjectInTargets
  - ICodeWarriorCreateObjectItem 122
- CreateProject
  - ICodeWarriorApp 37
- CreateProjectByFileSpec
  - ICodeWarriorApp 39
- CreateSubMenu
  - ICodeWarriorMenu 192
- CreateTarget
  - ICodeWarriorProject 221
- CreateTemporaryMenu
  - ICodeWarriorMenuManager 198
- CreateToolBar
  - ICodeWarriorWindow 388
- CreateUntitledFile
  - ICodeWarriorCreateFileItem 119
- CreateUserTree
  - ICodeWarriorApp 40
  - ICodeWarriorTarget 297



CWNativeXWindowPart Data Type 406  
 CWToolbarItemID Data Type 366  
 CWToolbarRegistryInfo Data Type 367

## D

### Data Type

Command Status 95  
 Creatable Item Category Constants 127  
 Creatable Items 127  
 CWNativeXWindowPart 406  
 CWToolbarIconRegistryInfo 367  
 Dialog Services 157  
 ECodeWarriorAccess 283  
 ECodeWarriorBuildOptions 249  
 ECodeWarriorCompileChoice 248  
 ECodeWarriorConvertOption 72  
 ECodeWarriorLinkFlags 145  
 ECodeWarriorProjectOption 73  
 ECodeWarriorRevertPanelOption 72  
 ECodeWarriorRunMode 249  
 ECodeWarriorSaveOption 73  
 ECodeWarriorShowSymbolLocation 283  
 ECodeWarriorTargetOutputKind 331  
 ECodeWarriorVCSCKIDState 378  
 ECodeWarriorVCSDBState 379  
 ECodeWarriorVCSInteractionOption 74  
 ECodeWarriorWhichTargetOptions 331  
 ECreateProjectType 128  
 EMsgType 214  
 ICodeWarriorToolbar  
   CWToolbarItemID 366  
 Menu Commands 95  
 SPopupMenuToolbarItem 366  
 SRegisterCommandGroup 96

### Data Types

Access Paths 30  
 Application 72  
 Designs 145  
 ICodeWarriorToolbar 366  
 Message 214  
 Project 248  
 Symbols 283  
 Targets 331  
 Windows 406

### DataModelCreated

ICodeWarriorAppEvents 65

### DataModelLoaded

ICodeWarriorAppEvents 66

### DataType

ECodeWarriorVCSFileLockState 379

### DeactivateEvent

ICodeWarriorWindowEvents 400

### DeleteMenuItem

ICodeWarriorMenu 194

### DeletingDesign

ICodeWarriorProjectEvents 241

### DesignClosing

ICodeWarriorDesignAttachment 141

### DesignCreated

ICodeWarriorProjectEvents 241

### DesignInitialized

ICodeWarriorDesignAttachment 141

### Designs API

Data Types 145

Overview 129

Reference 129

### DestroyCodeWarriorWindow

ICodeWarriorWindow 389

### Dialog Services API

Overview 147

Reference 151

Using 147

### Dialog Services Data Types 157

### Disconnect

ICodeWarriorVersionControl 371

### DoCommand

ICodeWarriorApp 41

### Documents API

Overview 159

Reference 159

### DrawItemRepresentation

ICodeWarriorCustomToolbarItem 345

## E

ECodeWarriorAccess Data Type 283

ECodeWarriorBuildOptions Data Type 249

ECodeWarriorCompileChoice Data Type 248

ECodeWarriorConvertOption Data Type 72

ECodeWarriorLinkFlags Data Type 145

ECodeWarriorProjectOption

Data Type 73

ECodeWarriorRevertPanelOption Data Type 72

ECodeWarriorRunMode Data Type 249

ECodeWarriorSaveOption

Data Type 73

ECodeWarriorShowSymbolLocation Data Type 283  
 ECodeWarriorTargetOutputKind Data Type 331  
 ECodeWarriorVCSCKIDState Data Type 378  
 ECodeWarriorVCSDBState Data Type 379  
 ECodeWarriorVCSFileLockState Data Type 379  
 ECodeWarriorVCSInteractionOption Data Type 74  
 ECodeWarriorWhichTargetOptions Data Type 331  
 ECreateProjectType Data Type 128  
 EMsgType Data Type 214  
 EPluginDataStorageLoc Data Type Data Type EPluginDataStorageLoc 248  
 Error Info API Overview 175 Reference 175  
 Execute ICodeWarriorDeferredAction 94  
 ExecuteCommand ICodeWarriorCommandHandler 89  
 ExpandGroup ICodeWarriorProjectDocument 169  
 Export ICodeWarriorProject 222  
 ExportByFileSpec ICodeWarriorProject 223  
**F**  
 FindAndAddFile ICodeWarriorDesign 135 ICodeWarriorTarget 298  
 FindAndAddFile2 ICodeWarriorDesign 136 ICodeWarriorTarget 299  
 FindAndAddFile2ByCollection ICodeWarriorTarget 300  
 FindAndAddFileByCollection ICodeWarriorTarget 301  
 FindClass ICodeWarriorSymbolContainer 278  
 FindClassInFile ICodeWarriorSymbolContainer 279  
 FindDataMemberByName ICodeWarriorClass 254

FindDesign ICodeWarriorProject 223  
 FindDesignForDataModel ICodeWarriorApp 41  
 FindFileByName ICodeWarriorProject 224  
 FindLogicalFolder ICodeWarriorApp 42  
 FindMethodByName ICodeWarriorClass 255  
 FindTarget ICodeWarriorProject 224  
 Folder Names, Standard 74  
**G**  
 Get ICodeWarriorVersionControl 371  
 get\_\_NewEnum Collection 80  
 get\_Access ICodeWarriorBaseClassInfo 252 ICodeWarriorDataMember 261 ICodeWarriorMethod 264  
 get\_AccessPathLocation ICodeWarriorAccessPath 14  
 get\_AccessPaths ICodeWarriorTarget 301  
 get\_AccessPathType ICodeWarriorAccessPath 15  
 get\_Action ICodeWarriorErrorInfo 176  
 get\_ActiveDocument ICodeWarriorApp 43 ICodeWarriorDocument 161  
 get\_AlwaysSearchUserPaths ICodeWarriorAccessPaths 22  
 get\_Application ICodeWarriorApp 43 ICodeWarriorProject 225  
 get\_BaseClass ICodeWarriorBaseClassInfo 252  
 get\_BaseClasses ICodeWarriorClass 255  
 get\_BrowserDB ICodeWarriorDesign 137 ICodeWarriorTarget 302  
 get\_BrowserEnabled

- ICodeWarriorTarget 302
- get\_BuildAgainst
  - ICodeWarriorSubProjectTarget 329
- get\_CanHaveMultipleEventSets
  - ICodeWarriorComponent 100
- get\_CKIDState
  - ICodeWarriorVCSState 375
- get\_Class
  - ICodeWarriorComponent 101
  - ICodeWarriorComponentEventSet 107
  - ICodeWarriorSymbol 274
- get\_ClassList
  - ICodeWarriorSymbolContainer 280
- get\_CompareInterface
  - ICodeWarriorApp 44
- get\_Container
  - ICodeWarriorSourceContext 274
  - ICodeWarriorSymbol 275
- get\_Count
  - Collection 80
- get\_CreatableItems
  - ICodeWarriorApp 44
- get\_DataModel
  - ICodeWarriorDesign 137
- get\_DBState
  - ICodeWarriorVCSState 376
- get\_Debugger
  - ICodeWarriorApp 44
- get\_DebugInfo
  - ICodeWarriorTargetFile 317
- get\_DeclarationLocation
  - ICodeWarriorSymbol 275
- get\_DefaultEvent
  - ICodeWarriorComponent 101
- get\_DefaultProject
  - ICodeWarriorApp 45
- get\_DefaultProjectDocument
  - ICodeWarriorApp 45
- get\_DefinitionCount
  - ICodeWarriorBuildMessages 202
- get\_DefinitionLocation
  - ICodeWarriorSymbol 276
- get\_Definitions
  - ICodeWarriorBuildMessages 203
- get\_Dependencies
  - ICodeWarriorTargetFile 318
- get\_Dependents
  - ICodeWarriorTargetFile 318
- get\_Design
  - ICodeWarriorTarget 303
- get\_Designs
  - ICodeWarriorProject 226
- get\_Dirty
  - ICodeWarriorDocument 161
- get\_Documents
  - ICodeWarriorApp 46
- get\_DWORDErr
  - ICodeWarriorErrorInfo 177
- get\_EndOffset
  - ICodeWarriorSourceContext 270
- get\_ErrorCount
  - ICodeWarriorBuildMessages 203
- get\_ErrorNumber
  - ICodeWarriorMessage 207
- get\_Errors
  - ICodeWarriorBuildMessages 204
- get\_EventConnectionsEnabled
  - ICodeWarriorComponent 102
- get\_Events
  - ICodeWarriorComponentProperty 107
- get\_EventSet
  - GetDefaultMethodName 105
- get\_EventSetName
  - ICodeWarriorComponentProperty 108
- get\_EventSets
  - ICodeWarriorComponent 102
- get\_FileLockState
  - ICodeWarriorVCSState 376
- get\_FileSpec
  - ICodeWarriorDocument 162
  - ICodeWarriorMessage 208
  - ICodeWarriorProject 226
  - ICodeWarriorProjectFile 246
  - ICodeWarriorSourceContext 271
  - ICodeWarriorTargetFile 319
  - ICodeWarriorTargetOutput 325
- get\_FullName
  - ICodeWarriorApp 46
- get\_FullPath
  - IFileSpec 186
- get\_Getter
  - ICodeWarriorComponentProperty 109
- get\_HasSelection
  - ICodeWarriorTextEngine 334

- get\_Height
  - ICodeWarriorDocument 162
- get\_HRESULT
  - ICodeWarriorErrorInfo 177
- get\_InformationCount
  - ICodeWarriorBuildMessages 204
- get\_Information
  - ICodeWarriorBuildMessages 205
- get\_InitBefore
  - ICodeWarriorTargetFile 319
- get\_IsAbstract
  - ICodeWarriorClass 257
  - ICodeWarriorMethod 265
- get\_IsConst
  - ICodeWarriorMethod 265
- get\_IsConstructor
  - ICodeWarriorMethod 266
- get\_IsDefined
  - ICodeWarriorSourceContext 271
- get\_IsDestructor
  - ICodeWarriorMethod 266
- get\_IsFinal
  - ICodeWarriorClass 257
  - ICodeWarriorDataMember 262
- get\_IsInline
  - ICodeWarriorMethod 267
- get\_IsNative
  - ICodeWarriorMethod 267
- get\_IsPublic
  - ICodeWarriorClass 258
- get\_IsStatic
  - ICodeWarriorDataMember 262
  - ICodeWarriorMethod 268
- get\_IsSynchronized
  - ICodeWarriorMethod 268
- get\_IsTransient
  - ICodeWarriorDataMember 263
- get\_IsVirtual
  - ICodeWarriorBaseClassInfo 253
  - ICodeWarriorMethod 269
- get\_IsVisible
  - ICodeWarriorProject 227
- get\_IsVolatile
  - ICodeWarriorDataMember 263
- get\_KeyName
  - ICodeWarriorUserTree 26
- get\_LineCount
  - ICodeWarriorTextEngine 335
- get\_LinkAgainst
  - ICodeWarriorSubProjectTarget 330
- get\_LinkAgainstOutput
  - ICodeWarriorSubTarget 327
- get\_MacOSErr
  - ICodeWarriorErrorInfo 178
- get\_MergeLibrary
  - ICodeWarriorTargetFile 320
- get\_MessageLength
  - ICodeWarriorMessage 208
- get\_MessageLineCount
  - ICodeWarriorMessage 209
- get\_MessageText
  - ICodeWarriorMessage 209
- get\_Method
  - GetDefaultMethodName 105
- get\_Methods
  - ICodeWarriorComponent 103
- get\_MWErr
  - ICodeWarriorErrorInfo 177
- get\_Name
  - GetDefaultMethodName 106
  - ICodeWarriorApp 47
  - ICodeWarriorComponentProperty 110
  - ICodeWarriorDesign 138
  - ICodeWarriorDocument 163
  - ICodeWarriorProject 228
  - ICodeWarriorProjectFile 246
  - ICodeWarriorSubProjectTarget 330
  - ICodeWarriorSymbol 276
  - ICodeWarriorTarget 303
  - ICodeWarriorTargetFile 320
  - ICodeWarriorUserTree 27
  - ICodeWarriorVersionControl 372
  - IFileSpec 186
- get\_OutputKind
  - ICodeWarriorTargetOutput 326
- get\_Path
  - ICodeWarriorAccessPath 15
- get\_Project
  - ICodeWarriorDesign 138
  - ICodeWarriorProjectAssociation 237
  - ICodeWarriorProjectDocument 169
  - ICodeWarriorProjectFile 246
  - ICodeWarriorTarget 304
- get\_ProjectFile
  - ICodeWarriorMessage 210

- 
- get\_ProjectFileCollection
    - ICodeWarriorTarget 305
  - get\_Projects
    - ICodeWarriorApp 48
  - get\_Properties
    - ICodeWarriorComponent 103
  - get\_ReadOnly
    - Collection 81
    - ICodeWarriorDocument 163
  - get\_Reason
    - ICodeWarriorErrorInfo 178
  - get\_Recursive
    - ICodeWarriorAccessPath 16
  - get\_SelectionEnd
    - ICodeWarriorTextEngine 336
  - get\_SelectionLineEnd
    - ICodeWarriorTextEngine 337
  - get\_SelectionLineStart
    - ICodeWarriorTextEngine 337
  - get\_SelectionStart
    - ICodeWarriorTextEngine 338
  - get\_SelectionText
    - ICodeWarriorTextEngine 338
  - get\_Setter
    - ICodeWarriorComponentProperty 110
  - get\_SimpleName
    - ICodeWarriorSymbol 276
  - get\_SourceLength
    - ICodeWarriorMessage 210
  - get\_SourceLineNumber
    - ICodeWarriorMessage 211
  - get\_SourceOffset
    - ICodeWarriorMessage 211
  - get\_StartOffset
    - ICodeWarriorSourceContext 272
  - get\_SubClasses
    - ICodeWarriorClass 260
  - get\_SubDirectories
    - ICodeWarriorAccessPath 16
  - get\_SubTargets
    - ICodeWarriorTarget 306
  - get\_SystemAccessPaths
    - ICodeWarriorAccessPaths 23
  - get\_Target
    - ICodeWarriorMessage 212
    - ICodeWarriorSubTarget 328
    - ICodeWarriorSymbolContainer 280
    - ICodeWarriorTargetFile 320
  - get\_TargetFileCollection
    - ICodeWarriorTarget 307
  - get\_Targets
    - ICodeWarriorDesign 139
    - ICodeWarriorProject 229
    - ICodeWarriorProjectFile 247
  - get\_TextEngine
    - ICodeWarriorTextDocument 171
  - get\_TextLength
    - ICodeWarriorTextEngine 339
  - get\_TokenLength
    - ICodeWarriorMessage 212
  - get\_TokenOffset
    - ICodeWarriorMessage 213
  - get\_Type
    - ICodeWarriorComponentProperty 111
    - ICodeWarriorMessage 213
    - ICodeWarriorUserTree 27
  - get\_UserAccessPaths
    - ICodeWarriorAccessPaths 23
  - get\_UserTree
    - ICodeWarriorAccessPath 17
  - get\_UserTrees
    - ICodeWarriorApp 49
    - ICodeWarriorTarget 308
  - get\_Value
    - ICodeWarriorUserTree 27
  - get\_VCSSState
    - ICodeWarriorProjectFile 247
  - get\_VersionControl
    - ICodeWarriorApp 49
    - ICodeWarriorProject 229
  - get\_Visible
    - ICodeWarriorApp 50
    - ICodeWarriorDocument 163
  - get\_WarningCount
    - ICodeWarriorBuildMessages 205
  - get\_WeakImport
    - ICodeWarriorTargetFile 321
  - get\_Width
    - ICodeWarriorDocument 164
  - get\_XPos
    - ICodeWarriorDocument 164
  - get\_YPos
    - ICodeWarriorDocument 165
  - GetCategory
    - ICodeWarriorCreatableItem 114
-

- GetCodeWarriorWindowSizeLocation
    - ICodeWarriorWindow 389
  - GetCommandStatus
    - ICodeWarriorCommandHandler 90
  - GetContainingDocument
    - ICodeWarriorToolbar 354
  - GetCreatedProjectType
    - ICodeWarriorCreateProjectItem 125
  - GetCurrentTarget
    - ICodeWarriorProject 225
  - GetDataMembers
    - ICodeWarriorClass 256
  - GetDataMembersWithAccess
    - ICodeWarriorClass 256
  - GetDefaultMethodName
    - get\_EventSet 105
    - get\_Method 105
    - get\_Name 106
    - GetDefaultMethodName 104
    - getDefaultMethodName 104
  - GetDefaultToolBarItems
    - ICodeWarriorWindowEvents 400
  - GetDisplayName
    - ICodeWarriorCreatableItem 115
  - GetHelpString
    - ICodeWarriorToolBarItemHelp 363
  - GetIcon
    - ICodeWarriorCreatableItem 115
  - GetIDEMainWindow
    - ICodeWarriorWindowManager 386
  - GetInitialState
    - ICodeWarriorPopupMenuToolBarItem 349
    - ICodeWarriorToggleButtonToolBarItem 352
  - GetItemRepresentationWidth
    - ICodeWarriorCustomToolBarItem 346
  - GetItemSizeInfo
    - ICodeWarriorCustomToolBarItem 346
  - GetItemWidth
    - ICodeWarriorPopupMenuToolBarItem 350
  - GetLineForOffset
    - ICodeWarriorTextEngine 335
  - GetLinkerName
    - ICodeWarriorTarget 303
  - GetMenusEnabledState
    - ICodeWarriorMenuManager 199
  - GetMethods
    - ICodeWarriorClass 258
  - GetMethodsWithAccess
    - ICodeWarriorClass 259
  - GetNamedPluginData
    - ICodeWarriorApp 47
    - ICodeWarriorProject 228
    - ICodeWarriorTarget 304
  - GetNativeWindowReference
    - ICodeWarriorWindow 390
  - GetNativeXWindowReference
    - ICodeWarriorWindow 390
  - GetOffsetForLine
    - ICodeWarriorTextEngine 336
  - GetProjectFileFromFileSpec
    - ICodeWarriorTarget 305
  - GetSampleTextString
    - ICodeWarriorPopupMenuToolBarItem 350
  - GetSetting
    - ICodeWarriorApp 48
  - GetSubProjects
    - ICodeWarriorTarget 306
  - GetTargetFileForProjectFile
    - ICodeWarriorTarget 307
  - GetTargetOutput
    - ICodeWarriorTarget 308
  - GetTextForLineRange
    - ICodeWarriorTextEngine 339
  - GetTextForOffsetRange
    - ICodeWarriorTextEngine 340
  - getting started 11
  - GetToolBarHeight
    - ICodeWarriorToolBar 354
  - GetToolBarItemText
    - ICodeWarriorToolBar 355
  - GetToolBarItemValue
    - ICodeWarriorToolBar 355
  - GetVCSSState
    - ICodeWarriorVersionControl 372
  - GetWindowToolBar
    - ICodeWarriorWindow 391
- 
- H**
- HandleMenuSelection
    - ICodeWarriorMenuHandler 196
  - HandlePopupSelection
    - ICodeWarriorPopupMenuToolBarItem 351
  - HasAttribute
    - ICodeWarriorWindow 391

- 
- I**
- ICodeWarriorAccessPath
    - get\_AccessPathLocation 14
    - get\_AccessPathType 15
    - get\_Path 15
    - get\_Recursive 16
    - get\_SubDirectories 16
    - get\_UserTree 17
    - put\_AccessPathLocation 17
    - put\_Recursive 18
  - ICodeWarriorAccessPaths
    - ApplyChanges 19
    - CreateAccessPath 19
    - CreateAccessPathByFileSpec 20
    - CreateAccessPathByPosition 21
    - get\_AlwaysSearchUserPaths 22
    - get\_SystemAccessPaths 23
    - get\_UserAccessPaths 23
    - put\_AlwaysSearchUserPaths 25
  - ICodeWarriorApp
    - AddCreatableItem 36
    - AddUserTree 36
    - AttemptModify 37
    - CreateProject 37
    - CreateProjectByFileSpec 39
    - CreateUserTree 40
    - DoCommand 41
    - FindDesignForDataModel 41
    - FindLogicalFolder 42
    - get\_ActiveDocument 43
    - get\_Application 43
    - get\_CompareInterface 44
    - get\_CreatableItems 44
    - get\_Debugger 44
    - get\_DefaultProject 45
    - get\_DefaultProjectDocument 45
    - get\_Documents 46
    - get\_FullName 46
    - get\_Name 47
    - get\_Projects 48
    - get\_UserTrees 49
    - get\_VersionControl 49
    - get\_Visible 50
    - GetNamedPluginData 47
    - GetSetting 48
    - ImportProject 50
    - ImportProjectByFileSpec 51
    - IsBuildInProgress 52
    - OpenDocument 52
    - OpenProject 53
    - OpenProjectByFileSpec 55
    - OpenProjectByFileSpecWithOptions 57
    - OpenProjectWithOptions 54
    - OpenTextDocument 59
    - OpenTextDocumentByFileSpec 59
    - OpenUntitledTextDocument 60
    - put\_AllowUserInteraction 61
    - put\_Visible 61
    - QueueDeferredAction 61
    - RemoveCreatableItem 62
    - RemoveNamedPluginData 63
    - RemoveUserTree 63
    - SetNamedPluginData 64
    - SetSetting 64
  - ICodeWarriorBaseClassInfo
    - get\_Access 252
    - get\_BaseClass 252
    - get\_IsVirtual 253
  - ICodeWarriorBaseClassInfo Interface 252
  - ICodeWarriorBuildMessages
    - get\_DefinitionCount 202
    - get\_Definitions 203
    - get\_ErrorCount 203
    - get\_Errors 204
    - get\_InformationCount 204
    - get\_Informations 205
    - get\_WarningCount 205
  - ICodeWarriorBuildMessages Interface 202
  - ICodeWarriorClass
    - FindDataMemberByName 254
    - FindMethodByName 255
    - get\_BaseClasses 255
    - get\_IsAbstract 257
    - get\_IsFinal 257
    - get\_IsPublic 258
    - get\_SubClasses 260
    - GetDataMembers 256
    - GetDataMembersWithAccess 256
    - GetMethods 258
    - GetMethodsWithAccess 259
  - ICodeWarriorClass Interface 254
  - ICodeWarriorCommandHandler
    - GetCommandStatus 90
    - ICodeWarriorCommandHandler 89
  - ICodeWarriorCommandHandler Interface
    - ICodeWarriorCommandHandler 89
-

- ICodeWarriorCommandRegistry
    - CreateNewCommandGroup 92
    - RegisterExternalCommand 93, 95
  - ICodeWarriorCommandRegistry Interface
    - ICodeWarriorCommandRegistry 92
  - ICodeWarriorCompare
    - CompareFiles 69
    - CompareFilesByFileSpec 70
    - CompareFolders 71
  - ICodeWarriorCompare Interface 69
  - ICodeWarriorComponent
    - get\_CanHaveMultipleEventSets 100
    - get\_Class 101
    - get\_DefaultEvent 101
    - get\_EventConnectionsEnabled 102
    - get\_EventSets 102
    - get\_Methods 103
    - get\_Properties 103
  - ICodeWarriorComponentEvent Interface 104
  - ICodeWarriorComponentEventSet
    - get\_Class 107
  - ICodeWarriorComponentEventSet Interface 107
  - ICodeWarriorComponentProperty
    - get\_Events 107
    - get\_EventSetName 108
    - get\_Getter 109
    - get\_Name 110
    - get\_Setter 110
    - get\_Type 111
  - ICodeWarriorComponentProperty Interface 109
  - ICodeWarriorCreatableItem
    - GetCategory 114
    - GetDisplayName 115
    - GetIcon 115
    - InvokesWizard 116
  - ICodeWarriorCreateFileItem
    - CanAddFileToProject 117
    - CanCreateUntitledFile 117
    - CreateAndAddFile 118
    - CreateUntitledFile 119
  - ICodeWarriorCreateFileItem Interface 117
  - ICodeWarriorCreateObjectItem
    - AreObjectsCreatedInDesign 120
    - CreateObjectInDesign 121
    - CreateObjectInTargets 122
    - NeedsObjectName 123
  - ICodeWarriorCreateObjectItem Interface 120
  - ICodeWarriorCreateProjectItem
    - CreateInExistingProject 124
    - CreateNewProject 125
    - GetCreatedProjectType 125
    - RequiresFileExtension 126
  - ICodeWarriorCreateProjectItem Interface 124
  - ICodeWarriorCustomToolBarItem
    - CreateItemControl 344
    - DrawItemRepresentation 345
    - GetItemRepresentationWidth 346
    - GetItemSizeInfo 346
  - ICodeWarriorCustomToolBarItem Interface 344
  - ICodeWarriorDataMember
    - get\_Access 261
    - get\_IsFinal 262
    - get\_IsStatic 262
    - get\_IsTransient 263
    - get\_IsVolatile 263
  - ICodeWarriorDataMember Interface 261
  - ICodeWarriorDeferredAction
    - Execute 94
  - ICodeWarriorDeferredAction Interface 94
  - ICodeWarriorDesign
    - AddAttachment 130
    - AddFile 131
    - AddFile2 131
    - AddFile2ByFileSpec 132
    - AddFileByFileSpec 133
    - CompileFiles 134
    - CompileFilesAndWaitToComplete 134
    - ContainsTarget 135
    - FindAndAddFile 135
    - FindAndAddFile2 136
    - get\_BrowserDB 137
    - get\_DataModel 137
    - get\_Name 138
    - get\_Project 138
    - get\_Targets 139
    - put\_Name 139
    - RemoveAttachment 139
    - RemoveTargetFromDesign 140
  - ICodeWarriorDesignAttachment
    - DesignClosing 141
    - DesignInitialized 141
    - RemovingAttachment 142
  - ICodeWarriorDesignAttachment Interface
    - ICodeWarriorDesignAttachment 141



- 
- ICodeWarriorDesignEvents
      - RemovingTarget 143
      - TargetAdded 144
    - ICodeWarriorDesignEvents Interface 143
    - ICodeWarriorDialogServices
      - NewItemDialog 152
      - OKCancelDialog 151
      - OKDialog 152
      - PostModalDialog 153
      - PreModalDialog 153
      - ReportErrorFromErrorInfo 154
      - SaveDontSaveDialog 154
      - SetPluginDialogCommandHandler 155
      - UpdatePluginDialogMenus 156
    - ICodeWarriorDocument
      - Activate 160
      - Close 161
      - get\_ActiveDocument 161
      - get\_Dirty 161
      - get\_FileSpec 162
      - get\_Height 162
      - get\_Name 163
      - get\_ReadOnly 163
      - get\_Visible 163
      - get\_Width 164
      - get\_XPos 164
      - get\_YPos 165
      - put\_Height 165
      - put\_Visible 165
      - put\_Width 166
      - put\_XPos 166
      - put\_YPos 167
      - Save 167
    - ICodeWarriorDocument Interface 160
    - ICodeWarriorErrorInfo
      - get\_Action 176
      - get\_DWORDErr 177
      - get\_HRESULT 177
      - get\_MacOSErr 178
      - get\_MWErr 177
      - get\_Reason 178
      - put\_Action 179
      - put\_DWORDErr 180
      - put\_HelpContext 180
      - put\_HelpFile 181
      - put\_HRESULT 180
      - put\_MacOSErr 182
      - put\_MWErr 181
      - put\_Reason 183
      - put\_Source 183
    - ICodeWarriorErrorInfo Interface 176
    - ICodeWarriorEvents
      - DataModelCreated 65
      - DataModelLoaded 66
      - ProjectVisible 67
      - QueryQuit 67
      - Quit 67
      - Startup 68
    - ICodeWarriorMenu
      - AppendMenuCommand 192
      - CreateSubMenu 192
      - DeleteMenuItem 194
      - RemoveItem 193
      - RenameMenuCommand 193
      - SetItemName 194
    - ICodeWarriorMenuHandler
      - HandleMenuSelection 196
      - UpdateMenuStatus 197
    - ICodeWarriorMenuHandler Interface 196
    - ICodeWarriorMenuManager
      - CreateTemporaryMenu 198
      - GetMenusEnabledState 199
      - SetMenusEnabledState 199
      - ShowCommandGroupMenu 200
    - ICodeWarriorMenuManager Interface 198
    - ICodeWarriorMessage
      - get\_ErrorNumber 207
      - get\_FileSpec 208
      - get\_MessageLength 208
      - get\_MessageLineCount 209
      - get\_MessageText 209
      - get\_ProjectFile 210
      - get\_SourceLength 210
      - get\_SourceLineNumber 211
      - get\_SourceOffset 211
      - get\_Target 212
      - get\_TokenLength 212
      - get\_TokenOffset 213
      - get\_Type 213
    - ICodeWarriorMessage Interface 207
    - ICodeWarriorMethod
      - get\_Access 264
      - get\_IsAbstract 265
      - get\_IsConst 265
      - get\_IsConstructor 266
      - get\_IsDestructor 266
      - get\_IsInline 267

- get\_IsNative 267
- get\_IsStatic 268
- get\_IsSynchronized 268
- get\_IsVirtual 269
- ICodeWarriorMethod Interface 264
- ICodeWarriorPopupMenuToolbarItem
  - BuildPopupMenuList 347
  - CleanupPopupMenuList 348
  - GetInitialState 349
  - GetItemWidth 350
  - GetSampleTextString 350
  - HandlePopupMenuSelection 351
- ICodeWarriorPopupMenuToolbarItem Interface 347
- ICodeWarriorProject
  - Build 216
  - BuildAndWaitToComplete 218
  - BuildAndWaitToCompleteWithOptions 218
  - BuildWithOptions 217
  - CloneTarget 219
  - Close 220
  - CompileFilesWithChoice 220
  - CreateDesign 221
  - CreateTarget 221
  - Export 222
  - ExportByFileSpec 223
  - FindDesign 223
  - FindFileByName 224
  - FindTarget 224
  - get\_Application 225
  - get\_Designs 226
  - get\_FileSpec 226
  - get\_IsVisible 227
  - get\_Name 228
  - get\_Targets 229
  - get\_VersionControl 229
  - GetCurrentTarget 225
  - GetNamedPluginData 228
  - RemoveDesign 230
  - RemoveDesignByName 230
  - RemoveFile 231
  - RemoveNamedPluginData 231
  - RemoveObjectCode 232
  - RemoveObjectCodeWithOptions 232
  - RemoveTarget 234
  - ReportMessage 235
  - SetCurrentTarget 235
  - SetNamedPluginData 236
  - SynchronizeStatus 236
- ICodeWarriorProject Interface 216
- ICodeWarriorProjectAssociation
  - get\_Project 237
  - put\_Project 238
- ICodeWarriorProjectAssociation Interface 237
- ICodeWarriorProjectDocument
  - CollapseGroup 168
  - ExpandGroup 169
  - get\_Project 169
  - SelectedFiles 170
  - SelectFiles 170
- ICodeWarriorProjectDocument Interface 168
- ICodeWarriorProjectEvents
  - BuildEnded 239
  - BuildStarted 240
  - DeletingDesign 241
  - DesignCreated 241
  - ProjectClosing 242
  - QueryAboutToBuild 242
  - QueryDeleteDesign 243
  - QueryUIClose 243
  - RevertCompleted 244
  - VisibleChanged 244
- ICodeWarriorProjectEvents Interface
  - ICodeWarriorProjectEvents 239
- ICodeWarriorProjectFile
  - CheckIn 245
  - CheckOut 245
  - get\_FileSpec 246
  - get\_Name 246
  - get\_Project 246
  - get\_Targets 247
  - get\_VCSState 247
- ICodeWarriorProjectFile Interface 245
- ICodeWarriorSourceContext
  - get\_Container 274
  - get\_EndOffset 270
  - get\_FileSpec 271
  - get\_IsDefined 271
  - get\_StartOffset 272
  - put\_EndOffset 272
  - put\_FileSpec 273
  - put\_StartOffset 273
- ICodeWarriorSourceContext Interface 270
- ICodeWarriorSubProjectTarget
  - get\_BuildAgainst 329
  - get\_LinkAgainst 330

- 
- get\_Name 330
  - ICodeWarriorSubProjectTarget Interface 329
  - ICodeWarriorSubTarget
    - get\_LinkAgainstOutput 327
    - get\_Target 328
  - ICodeWarriorSubTarget Interface 327
  - ICodeWarriorSymbol
    - get\_Class 274
    - get\_Container 275
    - get\_DeclarationLocation 275
    - get\_DefinitionLocation 276
    - get\_Name 276
    - get\_SimpleName 276
  - ICodeWarriorSymbol Interface 274
  - ICodeWarriorSymbolContainer
    - AddComponentAttachment 278
    - FindClass 278
    - FindClassInFile 279
    - get\_ClassList 280
    - get\_Target 280
    - RemoveComponentAttachment 281
    - ShowSymbolDeclaration 281
    - ShowSymbolDefinition 282
  - ICodeWarriorSymbolContainer Interface 278
  - ICodeWarriorTarget
    - AddFile 287
    - AddFile2 287
    - AddFile2ByFileSpec 288
    - AddFile2ByFileSpecification 289
    - AddFileByFileSpec 290
    - AddFileByFileSpecCollection 291
    - AddSubTarget 291
    - AddUserTree 292
    - Build 292
    - BuildAgainstSubProjectTarget 293
    - BuildAndWaitToComplete 293
    - BuildAndWaitToCompleteWithOptions 294
    - BuildWithOptions 294
    - CompileFiles 295
    - CompileFilesAndWaitToComplete 296
    - CompileFilesWithChoice 296
    - CreateUserTree 297
    - FindAndAddFile 298
    - FindAndAddFile2 299
    - FindAndAddFileByCollection 301
    - get\_AccessPaths 301
    - get\_BrowserDB 302
    - get\_BrowserEnabled 302
    - get\_Design 303
    - get\_Name 303
    - get\_Project 304
    - get\_ProjectFileCollection 305
    - get\_SubTargets 306
    - get\_TargetFileCollection 307
    - get\_UserTrees 308
    - GetLinkerName 303
    - GetNamedPluginData 304
    - GetProjectFileFromFileSpec 305
    - GetSubProjects 306
    - GetTargetFileForProjectFile 307
    - GetTargetOutput 308
    - LinkAgainstSubProjectTarget 310
    - LinkAgainstSubTarget 310
    - put\_BrowserEnabled 311
    - put\_Name 311
    - RemoveNamedPluginData 311
    - RemoveObjectCode 312
    - RemoveObjectCodeWithOptions 312
    - RemoveUserTree 314
    - SetNamedPluginData 315
    - SetupDebugging 315
    - SynchronizeStatus 316
  - ICodeWarriorTarget Interface 286
  - ICodeWarriorTargetFile
    - get\_DebugInfo 317
    - get\_Dependencies 318
    - get\_Dependents 318
    - get\_FileSpec 319
    - get\_InitBefore 319
    - get\_MergeLibrary 320
    - get\_Name 320
    - get\_Target 320
    - get\_WeakImport 321
    - put\_DebugInfo 321
    - put\_InitBefore 323
    - put\_MergeLibrary 323
    - put\_WeakImport 324
  - ICodeWarriorTargetFile Interface 317
  - ICodeWarriorTargetFindandAddFile2ByCollection 300
  - ICodeWarriorTargetOutput
    - get\_FileSpec 325
    - get\_OutputKind 326
  - ICodeWarriorTargetOutput Interface 325
  - ICodeWarriorTextDocument
    - get\_TextEngine 171
    - SaveACopyAs 172
    - SaveACopyAsByFileSpec 172

- SaveAs 173
- SaveAsByFileSpec 173
- ScrollToSelection 173
- ICodeWarriorTextDocument Interface 171
- ICodeWarriorTextEngine
  - get\_HasSelection 334
  - get\_LineCount 335
  - get\_SelectionEnd 336
  - get\_SelectionLineEnd 337
  - get\_SelectionLineStart 337
  - get\_SelectionStart 338
  - get\_SelectionText 338
  - get\_TextLength 339
  - GetLineForOffset 335
  - GetOffsetForLine 336
  - GetTextForLineRange 339
  - GetTextForOffsetRange 340
  - InsertText 340
  - put\_SelectionEnd 341
  - put\_SelectionLineEnd 341
  - put\_SelectionLineStart 341
  - put\_SelectionStart 342
  - put\_SelectionText 342
- ICodeWarriorTextEngine Interface 334
- ICodeWarriorToggleButtonToolbarItem
  - GetInitialState 352
  - StateChanged 353
- ICodeWarriorToggleButtonToolbarItem Interface 352
- ICodeWarriorToolbar
  - Data Types 366
  - GetContainingDocument 354
  - GetToolbarHeight 354
  - GetToolbarItemText 355
  - GetToolbarItemValue 355
  - IsToolbarVisible 356
  - ItemCreated 361
  - ItemDestroyed 362
  - ResetToolbarItem 356
  - SetToolbarItemEnabled 357
  - SetToolbarItemIcon 357
  - SetToolbarItemText 358
  - SetToolbarItemValue 359
  - ShowToolbar 359
  - Toolbar Constants 367
- ICodeWarriorToolbar Interface 354
- ICodeWarriorToolbarInstanceCreationNotification Interface 361
- ICodeWarriorToolbarItemHelp
  - GetHelpString 363
- ICodeWarriorToolbarItemHelp Interface 363
- ICodeWarriorToolbarItemRegistry
  - RegisterToolbarIcons 364
  - RegisterToolbarItem 365
- ICodeWarriorUserTree
  - get\_KeyName 26
  - get\_Name 27
  - get\_Type 27
  - get\_Value 27
  - put\_KeyName 28
  - put\_Name 28
  - put\_Type 28
  - put\_Value 29
- ICodeWarriorVCSFileStateListener Interface 377
- ICodeWarriorVCSState
  - get\_CKIDState 375
  - get\_DBState 376
  - get\_FileLockState 376
- ICodeWarriorVCSState Interface 375
- ICodeWarriorVCSStateListener
  - StateChanged 377
- ICodeWarriorVersionControl
  - CheckIn 370
  - CheckOut 370
  - Connect 371
  - Disconnect 371
  - Get 371
  - get\_Name 372
  - GetVCSState 372
  - IsConnected 373
  - UndoCheckOut 374
  - Unlock 373
- ICodeWarriorVersionControl Interface 370
- ICodeWarriorWindow
  - AssociateWindowWithProject 388
  - CreateToolbar 388
  - DestroyCodeWarriorWindow 389
  - GetCodeWarriorWindowSizeLocation 389
  - GetNativeWindowReference 390
  - GetNativeXWindowReference 390
  - GetWindowToolbar 391
  - HasAttribute 391
  - MoveCodeWarriorWindow 391
  - PutBehind 392
  - ReorderCodeWarriorWindow 393
  - SelectCodeWarriorWindow 397

- SetBackBrushes 393
- SetCodeWarriorWindowInitialBounds 394
- SetCodeWarriorWindowMinMaxSize 395
- SetCodeWarriorWindowTitle 395
- SetDialogColors 396
- SetEventHandler 396
- SetMaximumSleepTime 396
- ShowCodeWarriorWindow 397
- ICodeWarriorWindow Interface 387
- ICodeWarriorWindowEvents
  - ActivateEvent 399
  - DeactivateEvent 400
  - GetDefaultToolBarItems 400
  - Idle 400, 401
  - KeyDownEvent 401
  - MouseDownEvent 402
  - OkToClose 403
  - PreBeginUpdate 403
  - ToolBarSizeChange 403
  - UpdateEvent 404
  - WindowDestroyed 404
  - WindowResizedBy 405
- ICodeWarriorWindowManager
  - CenterWindow 383
  - CreateCodeWarriorWindow 384
  - GetIDEMainWindow 386
  - IsIDEInMDIMode 385
- ICodeWarriorWindowManager Interface 383
- ICodeWarriorApp Interface 35
- Idle
  - ICodeWarriorWindowEvents 400, 401
- IFileSpec
  - Clone 185
  - Copy 186
  - get\_FullPath 186
  - get\_Name 186
  - put\_FullPath 187
  - put\_Name 187
- IFileSpec Interface 185
- ImportProject
  - ICodeWarriorApp 50
- ImportProjectByFileSpec
  - ICodeWarriorApp 51
- InsertText
  - ICodeWarriorTextEngine 340
- installing
  - CodeWarrior 11
- Interface
  - ICodeWarriorAccessPath 14
  - ICodeWarriorAccessPaths 19
  - ICodeWarriorApp 35
  - ICodeWarriorBaseClassInfo 252
  - ICodeWarriorBuildMessages 202
  - ICodeWarriorClass 254
  - ICodeWarriorCompare 69
  - ICodeWarriorComponentEvent 104
  - ICodeWarriorComponentEventSet 107
  - ICodeWarriorComponentProperty 109
  - ICodeWarriorCreateFileItem 117
  - ICodeWarriorCreateObjectItem 120
  - ICodeWarriorCreateProjectItem 124
  - ICodeWarriorCustomToolBarItem 344
  - ICodeWarriorDataMember 261
  - ICodeWarriorDeferredAction 94
  - ICodeWarriorDesignEvents 143
  - ICodeWarriorDocument 160
  - ICodeWarriorErrorInfo 176
  - ICodeWarriorMenuHandler 196
  - ICodeWarriorMenuManager 198
  - ICodeWarriorMessage 207
  - ICodeWarriorMethod 264
  - ICodeWarriorPopupMenuToolBarItem 347
  - ICodeWarriorProject 216
  - ICodeWarriorProjectAssociation 237
  - ICodeWarriorProjectDocument 168
  - ICodeWarriorProjectFile 245
  - ICodeWarriorSourceContext 270
  - ICodeWarriorSubProjectTarget 329
  - ICodeWarriorSubTarget 327
  - ICodeWarriorSymbol 274
  - ICodeWarriorSymbolContainer 278
  - ICodeWarriorTarget 286
  - ICodeWarriorTargetFile 317
  - ICodeWarriorTargetOutput 325
  - ICodeWarriorTextDocument 171
  - ICodeWarriorTextEngine 334
  - ICodeWarriorToggleButtonToolBarItem 352
  - ICodeWarriorToolBar 354
  - ICodeWarriorToolBarInstanceCreationNotification 361
  - ICodeWarriorToolBarItemHelp 363
  - ICodeWarriorUserTree 26
  - ICodeWarriorVCSFileStateListener 377
  - ICodeWarriorVCSState 375
  - ICodeWarriorVersionControl 370
  - ICodeWarriorWindow 387
  - ICodeWarriorWindowManager 383

- IFileSpec 185
- InvokesWizard
  - ICodeWarriorCreatableItem 116
- IsBuildInProgress
  - ICodeWarriorApp 52
- IsConnected
  - ICodeWarriorVersionControl 373
- IsIDEInMDIMode
  - ICodeWarriorWindowManager 385
- IsToolbarVisible
  - ICodeWarriorToolbar 356
- Item
  - Collection 81
- ItemCreated
  - ICodeWarriorToolbar 361
- ItemDestroyed
  - ICodeWarriorToolbar 362

## K

- KeyDownEvent
  - ICodeWarriorWindowEvents 401

## L

- LinkAgainstSubProjectTarget
  - ICodeWarriorTarget 310
- LinkAgainstSubTarget
  - ICodeWarriorTarget 310

## M

- manual style 10
- Menu Commands Data Type 95
- Menus API
  - Overview 189
  - Reference 190
  - Using 189
- Message Data Types 214
- Messages API
  - Overview 201
  - Reference 201
- MouseDownEvent
  - ICodeWarriorWindowEvents 402
- MoveCodeWarriorWindow
  - ICodeWarriorWindow 391

## N

- NeedsObjectName

- ICodeWarriorCreateObjectItem 123
- new features 10
- NewItemDialog
  - ICodeWarriorDialogServices 152

## O

- OKCancelDialog
  - ICodeWarriorDialogServices 151
- OKDialog
  - ICodeWarriorDialogServices 152
- OkToClose
  - ICodeWarriorWindowEvents 403
- OpenDocument
  - ICodeWarriorApp 52
- OpenProject
  - ICodeWarriorApp 53
- OpenProjectByFileSpec
  - ICodeWarriorApp 55
- OpenProjectByFileSpecWithOptions
  - ICodeWarriorApp 57
- OpenProjectWithOptions
  - ICodeWarriorApp 54
- OpenTextDocument
  - ICodeWarriorApp 59
- OpenTextDocumentByFileSpec
  - ICodeWarriorApp 59
- OpenUntitledTextDocument
  - ICodeWarriorApp 60
- Overview
  - Collections API 77
  - Commands API 83
  - Components API 99
  - Designs API 129
  - Dialog Services API 147
  - Documents API 159
  - Error Info API 175
  - Menus API 189
  - Messages API 201
  - Projects API 215
  - Toolbar API 343
  - Windows API 381

## P

- Paths, Access 13
  - API Reference 13
  - ICodeWarriorAccessPath Interface 14
  - ICodeWarriorAccessPaths Interface 19

- ICodeWarriorUserTree Interface 26
- PostModalDialog
  - ICodeWarriorDialogServices 153
- PreBeginUpdate
  - ICodeWarriorWindowEvents 403
- PreModalDialog
  - ICodeWarriorDialogServices 153
- Project Data Types 248
- ProjectClosing
  - ICodeWarriorProjectEvents 242
- ProjectOpened
  - ProjectVisible 66
- Projects API
  - Overview 215
  - Reference 215
- ProjectVisible
  - ICodeWarriorAppEvents 67
  - ProjectOpened 66
- put\_AccessPathLocation
  - ICodeWarriorAccessPath 17
- put\_Action
  - ICodeWarriorErrorInfo 179
- put\_AllowUserInteraction
  - ICodeWarriorApp 61
- put\_AlwaysSearchUserPaths
  - ICodeWarriorAccessPaths 25
- put\_BrowserEnabled
  - ICodeWarriorTarget 311
- put\_DebugInfo
  - ICodeWarriorTargetFile 321
- put\_DWORDErr
  - ICodeWarriorErrorInfo 180
- put\_EndOffset
  - ICodeWarriorSourceContext 272
- put\_FileSpec
  - ICodeWarriorSourceContext 273
- put\_FullPath
  - IFileSpec 187
- put\_Height
  - ICodeWarriorDocument 165
- put\_HelpContext
  - ICodeWarriorErrorInfo 180
- put\_HelpFile
  - ICodeWarriorErrorInfo 181
- put\_HRESULT
  - ICodeWarriorErrorInfo 180
- put\_InitBefore
  - ICodeWarriorTargetFile 323
- put\_KeyName
  - ICodeWarriorUserTree 28
- put\_MacOSErr
  - ICodeWarriorErrorInfo 182
- put\_MergeLibrary
  - ICodeWarriorTargetFile 323
- put\_MWErr
  - ICodeWarriorErrorInfo 181
- put\_Name
  - ICodeWarriorDesign 139
  - ICodeWarriorTarget 311
  - ICodeWarriorUserTree 28
  - IFileSpec 187
- put\_Project
  - ICodeWarriorProjectAssociation 238
- put\_Reason
  - ICodeWarriorErrorInfo 183
- put\_Recursive
  - ICodeWarriorAccessPath 18
- put\_SelectionEnd
  - ICodeWarriorTextEngine 341
- put\_SelectionLineEnd
  - ICodeWarriorTextEngine 341
- put\_SelectionLineStart
  - ICodeWarriorTextEngine 341
- put\_SelectionStart
  - ICodeWarriorTextEngine 342
- put\_SelectionText
  - ICodeWarriorTextEngine 342
- put\_Source
  - ICodeWarriorErrorInfo 183
- put\_StartOffset
  - ICodeWarriorSourceContext 273
- put\_Type
  - ICodeWarriorUserTree 28
- put\_Value
  - ICodeWarriorUserTree 29
- put\_Visible
  - ICodeWarriorApp 61
  - ICodeWarriorDocument 165
- put\_WeakImport
  - ICodeWarriorTargetFile 324
- put\_Width
  - ICodeWarriorDocument 166
- put\_XPos
  - ICodeWarriorDocument 166

put\_YPos  
     IcodeWarriorDocument 167  
 PutBehind  
     IcodeWarriorWindow 392

**Q**

QueryAboutToBuild  
     IcodeWarriorProjectEvents 242  
 QueryDeleteDesign  
     IcodeWarriorProjectEvents 243  
 QueryQuit  
     IcodeWarriorAppEvents 67  
 QueryUIClose  
     IcodeWarriorProjectEvents 243  
 QueueDeferredAction  
     IcodeWarriorApp 61  
 QuickStart 11  
 Quit  
     IcodeWarriorAppEvents 67

**R**

RegisterExternalCommand  
     IcodeWarriorCommandRegistry 93, 95  
 RegisterToolBarIcons  
     IcodeWarriorToolBarItemRegistry 364  
 RegisterToolBarItem  
     IcodeWarriorToolBarItemRegistry 365  
 release notes 10  
 Remove  
     Collection 82  
 RemoveAttachment  
     IcodeWarriorDesign 139  
 RemoveComponentAttachment  
     IcodeWarriorSymbolContainer 281  
 RemoveCreatableItem  
     IcodeWarriorApp 62  
 RemoveDesign  
     IcodeWarriorProject 230  
 RemoveDesignByName  
     IcodeWarriorProject 230  
 RemoveFile  
     IcodeWarriorProject 231  
 RemoveItem  
     IcodeWarriorMenu 193  
 RemoveNamedPluginData  
     IcodeWarriorApp 63  
     IcodeWarriorProject 231

    IcodeWarriorTarget 311  
 RemoveObjectCode  
     IcodeWarriorProject 232  
     IcodeWarriorTarget 312  
 RemoveObjectCodeWithOptions  
     IcodeWarriorProject 232  
     IcodeWarriorTarget 312  
 RemoveTarget  
     IcodeWarriorProject 234  
 RemoveTargetFromDesign  
     IcodeWarriorDesign 140  
 RemoveUserTree  
     IcodeWarriorApp 63  
     IcodeWarriorTarget 314  
 RemovingAttachment  
     IcodeWarriorDesignAttachment 142  
 RemovingTarget  
     IcodeWarriorDesignEvents 143  
 RenameMenuCommand  
     IcodeWarriorMenu 193  
 ReorderCodeWarriorWindow  
     IcodeWarriorWindow 393  
 ReportErrorFromErrorInfo  
     IcodeWarriorDialogServices 154  
 ReportMessage  
     IcodeWarriorProject 235  
 RequiresFileExtension  
     IcodeWarriorCreateProjectItem 126  
 ResetToolBarItem  
     IcodeWarriorToolBar 356  
 RevertCompleted  
     IcodeWarriorProjectEvents 244

**S**

Save  
     IcodeWarriorDocument 167  
 SaveACopyAs  
     IcodeWarriorTextDocument 172  
 SaveACopyAsByFileSpec  
     IcodeWarriorTextDocument 172  
 SaveAs  
     IcodeWarriorTextDocument 173  
 SaveAsByFileSpec  
     IcodeWarriorTextDocument 173  
 SaveDontSaveDialog  
     IcodeWarriorDialogServices 154  
 ScrollToSelection 173



- 
- SelectCodeWarriorWindow
    - ICodeWarriorWindow 397
  - SelectedFiles
    - ICodeWarriorProjectDocument 170
  - SelectFiles
    - ICodeWarriorProjectDocument 170
  - SetBackBrushes
    - ICodeWarriorWindow 393
  - SetCodeWarriorWindowInitialBounds
    - ICodeWarriorWindow 394
  - SetCodeWarriorWindowMinMaxSize
    - ICodeWarriorWindow 395
  - SetCodeWarriorWindowTitle
    - ICodeWarriorWindow 395
  - SetCurrentTarget
    - ICodeWarriorProject 235
  - SetDialogColors
    - ICodeWarriorWindow 396
  - SetEventHandler
    - ICodeWarriorWindow 396
  - SetItemName
    - ICodeWarriorMenu 194
  - SetMaximumSleepTime
    - ICodeWarriorWindow 396
  - SetMenusEnabledState
    - ICodeWarriorMenuManager 199
  - SetNamedPluginData
    - ICodeWarriorApp 64
    - ICodeWarriorProject 236
    - ICodeWarriorTarget 315
  - SetPluginDialogCommandHandler
    - ICodeWarriorDialogServices 155
  - SetSetting
    - ICodeWarriorApp 64
  - SetToolbarItemEnabled
    - ICodeWarriorToolbar 357
  - SetToolbarItemIcon
    - ICodeWarriorToolbar 357
  - SetToolbarItemText
    - ICodeWarriorToolbar 358
  - SetToolbarItemValue
    - ICodeWarriorToolbar 359
  - SetupDebugging
    - ICodeWarriorTarget 315
  - ShowCodeWarriorWindow
    - ICodeWarriorWindow 397
  - ShowCommandGroupMenu
    - ICodeWarriorMenuManager 200
  - ShowSymbolDeclaration
    - ICodeWarriorSymbolContainer 281
  - ShowSymbolDefinition
    - ICodeWarriorSymbolContainer 282
  - ShowToolbar
    - ICodeWarriorToolbar 359
  - SPopupMenuItem Data Type 366
  - SRegisterCommandGroup Data Type 96
  - Standard Folder Names 74
  - Startup
    - ICodeWarriorAppEvents 68
  - StateChanged
    - ICodeWarriorToggleButtonToolbarItem 353
    - ICodeWarriorVCSStateListener 377
  - Symbols Data Types 283
  - SynchronizeStatus
    - ICodeWarriorProject 236
    - ICodeWarriorTarget 316
- ## T
- TargetAdded
    - ICodeWarriorDesignEvents 144
  - Targets Data Types 331
  - Text API
    - Overview
      - Overview
      - Text API 333
    - Reference 333
  - Toolbar API
    - Overview 343
    - Reference 343
  - Toolbar Constants
    - ICodeWarriorToolbar 367
  - ToolbarSizeChange
    - ICodeWarriorWindowEvents 403
  - typographical conventions 10
- ## U
- UndoCheckOut
    - ICodeWarriorVersionControl 374
  - Unlock
    - ICodeWarriorVersionControl 373
  - UpdateEvent
    - ICodeWarriorWindowEvents 404
  - UpdateMenuStatus 197
-

## Index

---

- UpdatePluginDialogMenus
  - ICodeWarriorDialogServices 156
- UpdatePort
  - ICodeWarriorWindow'ICodeWarriorWindow
    - UpdatePort 398
- Using
  - Collections API
    - Collections API
      - Using 77
  - Commands API 84
  - Dialog Services API 147
  - Menus API 189
  - Windows API 381

## V

- Version Control API Reference 369
- VisibleChanged
  - ICodeWarriorProjectEvents 244

## W

- WindowDestroyed
  - ICodeWarriorWindowEvents 404
- WindowResizedBy
  - ICodeWarriorWindowEvents 405
- Windows API
  - Overview 381
  - Reference 382
  - Using 381
- Windows Data Types 406