# Understand User Guide and Reference Manual

Version 4.0 October 2015



Scientific Toolworks, Inc. 53 N Main St. George, UT 84770

Copyright © 2015 Scientific Toolworks, Inc. All rights reserved.

The information in this document is subject to change without notice. Scientific Toolworks, Inc. makes no warranty of any kind regarding this material and assumes no responsibility for any errors that may appear in this document.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 (48 CFR). Contractor/Manufacturer is Scientific Toolworks, Inc., 53 N. Main, St. George, UT 84770.

NOTICE: Notwithstanding any other lease or license agreement that may pertain to or accompany the delivery of this restricted computer software, the rights of the Government regarding use, reproduction, and disclosure are as set forth in subparagraph (c)(1) and (2) of Commercial Computer Software-Restricted Rights clause at FAR 52.227-19.

Part Number: USTAND4.0-GEN-UG-802 (10/15)

# Contents

Chapter 1	Introduction What is Understand? Licensing Issues Languages Supported For Those Who Don't Like to Read Manuals	13 14 15 16
Chapter 2	Parts and Terminology	18
	Understand Terminology Parts	19 20
	Starting Understand Other Ways to Run Understand	21 22
	Context Menus Are Everywhere	23
	Quickly Find Things in Your Source	25
	Entity Filter	25
	Entity Locator	26
	Find in Files	20
	Favorites	27
	Information Browser	28
	Source Editor	29
	Architecture Browser	30
	Graphical Views	31
	ASCII and HTML Reports	32
	APIs for Custom Reporting	32
Chapter 3	Configuring Your Project	
	About Understand Projects	34
	The Understand Project Database	34
	Creating a New Project New Project Wizard	35 35
	Project Configuration Dialog	39
	Languages Category	41
	Files Category	42 43
	Adding Files	44
	Removing Directories and Files	44
	Scanning Watched Directories	45 46
	Setting File Portability	47
	File Types	48
	File Options	49

Scheduled Activities	50
Metrics	52
	53
	54 54
Reports > Output Category	55
Reports > Selected Category	56
Visual Studio.	57
Annotations	59
Ada Options	61
Áda > Macros Category	63
Assembly Options	65
COBOL Options	66
COBOL > Copybooks Category	67
C++ (Fuzzy) Options	68
C++ > Includes Category	70
C++ > Includes > Auto Category	71
C++ > Includes > Ignore Category	71
C++ > Includes > Replacement Text	72
$C_{++} > Macros > Undefines Category$	74
C++ (Strict) Ontions	75
C++ (Strict) > Includes Category	77
C++ (Strict) > Includes > Frameworks Category	78
C++ (Strict) > Includes > Prefix Headers Category	79
C++ (Strict) > Macros Category	79
C# Options	79
Fortran Options	81
Fortran>Includes Category	83
Other Fortran Categories	83
Java Options	84
Java > Class Paths Category	84
Eclipse Plugin	85
JOVIAL Options	86
Jovial > !Copy Category	87
Pascal Options	88
Pascal > Macros Category	89
Pascal > Namespaces Calegory	00
Pascal > Standard Library Paths Category	89 80
Pascal > Standard Library Paths Category Pascal > Search Paths Category	89 89 90
Pascal > Standard Library Paths Category Pascal > Search Paths Category	89 89 90
Pascal > Standard Library Paths Category Pascal > Search Paths Category PL/M Options PL/M>Includes Category	89 89 90 90 91
Pascal > Standard Library Paths Category Pascal > Search Paths Category PL/M Options PL/M>Includes Category Python Options	89 89 90 90 91 92
Pascal > Standard Library Paths Category Pascal > Search Paths Category PL/M Options PL/M>Includes Category Python Options Python > Imports Category	89 89 90 91 92 92
Pascal > Standard Library Paths Category Pascal > Search Paths Category PL/M Options PL/M>Includes Category Python Options Python > Imports Category VHDL Options	89 89 90 91 92 92 93
Pascal > Standard Library Paths Category Pascal > Search Paths Category PL/M Options PL/M>Includes Category Python Options Python > Imports Category VHDL Options Web Options	<ul> <li>89</li> <li>89</li> <li>90</li> <li>90</li> <li>91</li> <li>92</li> <li>92</li> <li>93</li> <li>93</li> </ul>

	General Category96User Interface Category98User Interface > Lists Category99User Interface > Alerts Category100User Interface > Windows Category101User Interface > Application Styles Category103Key Bindings Category104Analyze Category106Configure Category106Command Window Category106Portability Category107Dependency Category108Editor Category108Editor > Advanced Category114Editor > Styles Category114Editor > Navigation Category116
	Editor > External Editor Category
	Graphs Category 118
	Analyzing the Code
	Lising the Missing Header Files Tool 122
	Using the Undefined Macros Tool
Chapter 4	Exploring Your Codebase PLEASE RIGHT-CLICK
	Various Windows Explained 128
	Entity Filter
	Information Browser
	Displaying More or Less Information 132
	Searching the Information Browser
	Syncing the Information Browser
	Visiting Source Code
	Visiting References 135
	Viewing Metrics       136         Saving and Printing Information Browser Text       136         Entity History       136
	Project Browser 137
	Exploring a Hierarchy 139
	Dependency Browser 140
	Favorites
	Creating a Favorite Entity
	Creating a Favorite View
	Using a Favorites Group

Chapter 5	Searching Your Source	
	Searching: An Overview	147
	Instant Search.	148
	Find in Files	150
	Find Results	152
	Replace in Files	153
	Entity Locator	155
	Resizing Columns	155
	Long versus Short Names	155
	Column Headers	156
	Filtering the List	157
	Finding Windows	160
	Source Visiting History	161
	View Menu Commands	161
	Displaying Toolbars	162
	Searching in a File	163
	Find Next and Previous	163
	Find & Replace.	163
	Contextual Information Sidebar	164
Chapter 6	Editing Your Source	
	Source Editor	166
	Scope List	167
		167
	Status Line	167
	Browse Mode	168
	Context Menu.	169
	Hover Text	170
	Saving Source Code	170
	Refactoring Tools	171
	Renaming Entities	172
		173
		174
	Extract Temp	175
		177
	Previewer	177
	Bracket Matching	178
	Folding and Hiding	178
	Splitting the Editor Window	179
	Commenting and Uncommenting.	179
	Unanging Uase	179 120
		180
	Insert and Overtype Modes	180
	Sorting Lines Alphabetically	180

	Keyboard CommandsRecording, Playing, and Saving MacrosCreating and Opening FilesBookmarkingManaging Source Editor TabsChanging the Source Code Font Size	180 180 181 181 183 183
	Annotations . Adding an Annotation . Editing an Annotation . Deleting an Annotation . Managing Annotation Files and Display . Searching Annotations . Filtering Annotations . Managing Orphaned Annotations .	184 184 186 186 186 187 188 189
	Printing Source Views	190
Chapter 7	Architecting Your Codebase	
	About Architectures	192
	Using the Architecture Browser	193 193
	Viewing Architecture Dependency Crente	100
		190
	Graph Architecture View	199
	Viewing Architecture Metrics	200
	Managing Architectures	201
	Creating an Architecture	202
	Using the Architecture Wizard	203
	Editing an Architecture	204
	Using XML to Manage Architectures	206
	Exporting Architectures to XML	206
	Importing XML Architectures	206
Chapter 8	Using Reports	
	Configuring Reports	208
	Customizing Report Colors	210
	Generating Reports	210
	Viewing Reports	211
	An Overview of Report Categories	212 213
	Cross-Reference Reports	214
	Data Dictionary Report	214
	Program Unit Cross-Reference Report	215
	File Contents Report	215
	Ubject Gross-Reference Report	216
	Class and Interface Cross-Reference	216
	Macro Cross-Reference	217
	Include File Cross-Reference.	217

	Exception Cross-Reference Report	218
	Structure Reports	219
	Declaration Tree.	219
	Extend Tree	220
	Invocation Tree Report.	220
	Simple Invocation Tree Report.	220
	With Tree Report	221
	Simple With Tree Report	221
		221
	Import Report	221
	Quality Reports	221
	Program Unit Complexity Report	222
	Fortran Extension Usage Report	223
	Implicitly Declared Objects Report	223
	Uninitialized Items	224
	Unused Objects and Functions	224
	Unused Objects Report	224
	Unused Types Report	225
	Unused Program Units Report	225
	Uses Not Needed Report	225
	Withs Not Needed Report	225
	Metrics Reports	226
		226
		221
	Class OO Matrice Depart	222
	Class OO Metrics Report	227 228
	Class OO Metrics Report	227 228 228
	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report	227 228 228 229
	Class OO Metrics Report	227 228 228 229
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report	227 228 228 229
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics	227 228 228 229 231
Chapter 9	Class OO Metrics Report Program Unit Metrics Report	227 228 228 229 231 232
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser	227 228 228 229 231 232 233
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML	227 228 228 229 231 232 233 234
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File	227 228 229 231 232 233 234 235
Chapter 9	Class OO Metrics Report         Program Unit Metrics Report         File Metrics Report         File Average Metrics Report         Using Metrics         About Metrics         Metrics Summary         Metrics Browser         Exporting Metrics to HTML         Exporting Metrics to a CSV File         Configuring Metric Charts	227 228 229 231 232 233 234 235 237
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Configuring Metric Charts Using the Metrics Treemap	227 228 229 231 232 233 234 235 237 240
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Configuring Metric Charts Using the Metrics Treemap Exporting Dependency Metrics	227 228 229 231 232 233 234 235 237 240 242
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Using the Metrics Treemap Exporting Dependency Metrics Exporting Dependencies to a CSV File	227 228 229 231 232 233 234 235 237 240 242 243
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report <b>Using Metrics</b> About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Using the Metrics Treemap Exporting Dependency Metrics Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File	227 228 229 231 232 233 234 235 237 240 242 243 244
Chapter 9	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Configuring Metrics Treemap. Exporting Dependencies to a CSV File Exporting Dependencies to a CSV Matrix File Exporting Dependencies to a CSV Matrix File	227 228 229 231 232 233 234 235 237 240 242 243 244 244
Chapter 9 Chapter 10	Class OO Metrics Report	227 228 229 231 232 233 234 235 237 240 242 243 244 244
Chapter 9 Chapter 10	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Configuring Metric Charts Using the Metrics Treemap Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File Exporting Dependencies to a CSV Matrix File Exporting Dependencies to Cytoscape Using Graphical Views Project Overview Graphics	227 228 229 231 232 233 234 235 237 240 242 243 244 244 244
Chapter 9 Chapter 10	Class OO Metrics Report Program Unit Metrics Report File Metrics Report File Average Metrics Report Using Metrics About Metrics Metrics Summary Metrics Browser Exporting Metrics to HTML Exporting Metrics to a CSV File Configuring Metric Charts Using the Metrics Treemap Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File Exporting Dependencies to a CSV File Exporting Dependencies to a CSV Matrix File Exporting Dependencies to Cytoscape Using Graphical Views Project Overview Graphics Graphical View Browsers	227 228 229 231 232 233 234 235 237 240 242 243 244 244 244 244
Chapter 9 Chapter 10	Class OO Metrics Report	227 228 229 231 232 233 234 235 237 240 242 243 244 244 244 244 244
Chapter 9 Chapter 10	Class OO Metrics Report	227 228 229 231 232 233 234 235 237 240 242 243 244 244 244 244 244 244

General Commands for Using Graphical Browsers	250
	252
	253
	253
	253
	204
	254
	255
	258
Structure View Examples	258
Graphical Notation	263
Controlling Graphical View Layout	263
Called by Menu	264
Comments Menu	264
Constants Menu	264
Default Members Menu	264
Dependent Of Menu	264
Dependent Menu	264
Depth	264
Duplicate Subtrees Menu	264
Expand Recursive Notes	265
Expand Repeated Notes	265
Extended By Menu	265
Extends Menu	265
External Functions Menu	265
Filename Menu	265
Function Pointer Menu	265
Globals Menu	265
	265
Implemented By Menu	265
	200
	200
	200
	200
	200
	200
	200
	200
	207
	268
	268
	268
Name Menu	268
	268
Operators Menu	268
Parameters Menu.	269
Private Members Menu	269
Protected Members Menu	269
Public Members Menu	269
Renames Menu	269
Routines Menu	269

	Scale MenuSort Menu.Spacing MenuSql MenuSql MenuStatic MenuText MenuTypes MenuTypetext MenuUnknown MenuUnresolved MenuUsedby MenuUses MenuVariables MenuWiths MenuWith Bys MenuControlling Cluster Graph LayoutCluster Control Flow GraphsSaving Views to FilesSaving Views as Visio FilesSaving Views as DOT Files	269 270 270 270 270 270 271 271 271 271 271 271 271 271 271 272 274 275 276 276 277
	Printing Graphical ViewsGraphical View Printing	278 278
Chapter 11	Using CodeCheck for Standards Verification	
	About CodeCheck. Running a CodeCheck Files Tab Checks Tab Exporting and Importing Configurations Viewing CodeCheck Results Using the Result Log Using the Results by File Tab Using the Results by Check Tab Using the Result Locator Using the Result Locator Using the Result Treemap Printing and Exporting Results. Ignoring Checks and Violations Using CodeCheck Configurations. Writing CodeCheck Scripts. Installing Custom Scripts	281 282 283 285 285 285 286 287 288 289 290 290 290 292 293 294
Chapter 12	Comparing Source Code Comparing Files and Folders Comparing Entities Comparing Text Exploring Differences Code Comparison	296 299 300 301 301

	Patch File	304 304
Chapter 13	Running Tools and External Commands	
	Configuring Tools Variables Adding Tools to the Context Menus Adding Tools to the Tools Menu Adding Tools to the Toolbar	306 308 313 314 315
	Importing and Exporting Tool Commands         Running External Commands         Using the Eclipse Plugin	316 317 318
Chapter 14	Command Line Processing	010
Chapter 15	Using the und Command Line	320 322 322 323 323 324 324 324 325 325 326 326 326 326 327 328 329
Chapter 15	QUICK Reference         File Menu         Edit Menu         Search Menu         View Menu         Project Menu         Project Menu         Reports Menu         Metrics Menu         Graphs Menu         CodeCheck Menu         Annotations Menu         Tools Menu         Window Menu         Help Menu	331 332 333 333 334 334 335 335 335 335 336 336 337

# Chapter 1 Introduction

This chapter introduces the Understand software.

This manual assumes a moderate understanding of the programming language in which your project is written.

This chapter contains the following sections:

Section	Page
What is Understand?	13
Licensing Issues	14
Languages Supported	15
For Those Who Don't Like to Read Manuals	16

### What is Understand?

*Understand* is a static analysis tool focused on source code comprehension, metrics, and standards testing. It is designed to help maintain and understand large amounts of legacy or newly created source code. It provides a cross-platform, multi-language, maintenance-oriented IDE (interactive development environment).

The source code analyzed may include C, C++, C#, Objective C/Objective C++, Ada, Java, Pascal/Delphi, COBOL, JOVIAL, VHDL, Fortran, PL/M, Python, PHP, HTML, CSS, JavaScript, and XML.

It offers code navigation using a detailed cross-referencing, a syntax-colorizing "smart" editor, and a variety of graphical reverse engineering views.



*Understand* creates a repository of the relations and structures contained within the software project. The repository is then used to learn about the source code.

Understand has analysis features that help you quickly answer questions such as:

- · What is this entity?
- Where is it changed?
- Where is it referenced?
- Who depends on it?
- What does it depend on?

*Understand* has architecture features that help you create hierarchical aggregations of source code units. You can name these units and manipulate them in various ways to create interesting hierarchies for analysis.

### **Licensing** Issues

To view or change the license being used, choose **Help->Licensing** from the menus, select the license you want to use, and then restart *Understand*.

P	Licensing			? ×		
	Product Licensing Please select the lice	nse you wish to use				
	Languages	Location	Quantity			
	Any	Single Developer (Local)	1 Available			
	Delete All Licenses	Add a Licens	Show Users			
F	For more information, or licensing support please visit our <u>web site</u> , or <u>email support</u> .       OK    Cancel					

If you have multiple licenses, you can select the one you want to use here. If you have a new license key, click **Add a License** and choose to add either an evaluation or Single Developer License (SDL) or the name of a license server.

If you are using a floating license, you can check the **Show Users** box to see the currently active users. Click the double-arrow icon to refresh the license use information.

See the Scientific Toolworks website for additional information about installation and licensing.

### Languages Supported

The following list provides a brief overview of the language versions and/or compilers that *Understand* supports:

- Ada: Understand supports Ada83, Ada95, and Ada05 code, separately, or in combination.
- Assembly: Assembly code for Freescale Coldfire microprocessors and the Motorola 680000 (68K) family is supported.
- C/C++: Understand analyzes K&R or ANSI C source code and most constructs of the C++ language. Understand works with any C compiler, and has been tested with most of the popular ones.
- **Objective C/Objective C++:** Understand provides a strict analyzer option that supports these languages.
- C#: Understand supports C#.
- **COBOL:** *Understand* supports the Ansi85, Micro Focus, AcuCobol, and IBM compilers.
- Fortran: Understand supports FORTRAN 77, Fortran 90, Fortran 95, and Fortran 2003, in both free and fixed format. Extensions supported include Harris Fortran and DEC Fortran. We often expand Understand to support common compiler extensions. If you find that the compiler extensions you are using are not currently supported, contact us at support@scitools.com.
- Java: Understand supports most of JDK 1.3, 1.4, 5, and 6. Specifically, the generics introduced in JDK 5 are not currently supported. Source code containing generics may be analyzed but generics information will be ignored.
- JOVIAL: JOVIAL73 and JOVIAL3 are supported.
- **Pascal:** Understand supports all versions of Borland's Delphi language and Borland's Turbo Pascal language. It also supports ISO 7185: 1990 (also known as Unextended Pascal) with DEC Pascal extensions. You can also enable support for Ingres embedded SQL statements.
- PL/M: The standard version for PL/M 80/86 is supported.
- Python: Understand supports both Python 2.x and 3.x.
- VHDL: Versions VHDL-87, VHDL-93, and VHDL-2001 are supported.
- Web: HTML, PHP, CSS, JavaScript, and XML files are supported.

For information about support for a specific language syntax, search the Build Log on the Scientific Toolworks website (www.scitools.com/support/buildLogs.php).

### For Those Who Don't Like to Read Manuals

If you are like many engineers at Scientific Toolworks, you like to just dig in and get going with software. We encourage that, or at least we are pragmatic enough to know you will do it anyway! So feel free to use this manual as a safety net, or to find the less obvious features. However, before you depart the manual, skim the next chapter for tips on effectively utilizing what *Understand* has to offer.

Here are some places other than this manual to look for advice:

- Use the links in the Getting Started display (Help > Getting Started from the menus)
- Choose Help > Help Content from the menus.
- Use Help > Example Projects to play with sample code.
- Choose Help > Frequently Asked Questions to see the FAQ list on our website.
- Choose Help > View SciTools Blog to read the blog on our website. You can get the latest blog topics by clicking the Refresh button.
- For information about various features, see www.scitools.com/support.

For more advanced users, try these information sources:

- Choose Help > About Understand to see which build you are currently running.
- See www.scitools.com/support/buildLogs.php to search through the build logs. Use the link on that page to "Sign up to receive via Email" new build notes and build announcements.
- Choose Help > Key Bindings for keystroke help.
- See scitools.com/support/metrics\_list/ for details about specific metrics.
- Choose Help > Perl API Documentation and Help > Python API Documentation for help on scripting. Java API documentation is provided in the doc/manuals/java subdirectory of the Understand installation.

# Chapter 2 Parts and Terminology

This chapter helps you put *Understand* to good use quickly and easily by describing the basic windows in *Understand*.

This chapter contains the following sections:

Section	Page
Using Understand Windows	18
Understand Terminology	19
Starting Understand	21
Context Menus Are Everywhere	23
Quickly Find Things in Your Source	25
Information Browser	28
Source Editor	29
Architecture Browser	30
Graphical Views	31
ASCII and HTML Reports	32
APIs for Custom Reporting	32

### **Using Understand Windows**

*Understand* has a main window and many smaller areas that open within the *Understand* application window. You can arrange these areas in your workspace to meet your needs.



- **Title Bar:** You can drag the title bar of an area around the main window. If you move to the edge of the main window, a docking area expands. If you drop the area there, it "docks" to the edge of the main window.
- **Pushpins and Drawers:** Click the same edge of the main window to which this area was docked. This is a "drawer" that opens automatically if you point your mouse at the tab title. The drawer closes if you move your mouse away from the area without clicking on it or if you click the title tab of the currently open drawer.

Click the icon to "pin" a drawer open. Pinned drawers have a title bar and title bar icons like the ones shown above.

- **Dock/Undock:** Click the 🗗 icon to change the area to an undocked window. Click the icon again in an undocked window to return to a docked area.
- Close: Click the "X" icon to close the area or undocked window.
- **Drop-down:** Click the icon to see the context menu for this area. Right-clicking an item within an area usually displays a context menu specific to that item.
- Sliding Frame: You can drag the frames between window areas to change their sizes.
- **Previous and Next:** Each area type has different icons below the title bar. For the Information Browser area shown, you can browse through the history of entities viewed. For other areas, you will see other icons.

### **Understand Terminology**

Before continuing with the rest of this manual, please take a moment to familiarize yourself with *Understand*'s terminology. Doing so will make reading the manual more helpful and put you on the same sheet of music as the technical support team should you need to email or call.

**Architecture:** An architecture is a hierarchical aggregation of source code units (entities). An architecture can be user created or automatically generated. Architectures need not be complete (that is, an architecture's flattened expansion need not reference every source entity in the database), nor unique (that is, an architecture's flattened expansion need not maintain the set property).

**Database:** The database is where the results of the source code analysis, as well as project settings, are stored. By default, this is a project's ".udb" file.

**Entity:** An *Understand* "entity" is anything it has information about. In practice this means anything declared or used in your source code and the files that contain the project. Subroutines, variables, and source files are all examples of entities.

**Project:** The set of source code you have analyzed and the settings and parameters chosen. A "project file" contains the list of source files and the project settings.

**Relationship:** A particular way that entities relate to one another. The names of relationships come from the syntax and semantics of a programming language. For instance, subroutine entities can have "Call" relationships and "CalledBy" relationships.

**Script:** Generally a Perl script. These can be run from within *Understand*'s GUI, or externally via the "uperl" command. The *Understand* Perl API provides easy and direct access to all information stored in an Understand database.

 Parts
 The following figure shows some commonly used main parts of the Understand graphical user interface (GUI):



### **Starting Understand**

When you install *Understand* on Windows, a command to launch the software is added to your Windows **Start** menu in the SciTools folder.

When you start *Understand*, you see the **Getting Started** tab in the Understand window. To begin creating a new project, click **New Project...** and see *Creating a New Project* on page 35 for details.

If you've used a project recently, it is listed in the Getting Started tab, and you can click to open it. If the existing project you want to open isn't listed, click **Open Project...** and browse for it.

SciToes Understand Source Code Analysis & Metrics	
Recent Projects	News & Announcements Refresh
C:\Users\Yvonne\AppData\Roaming\SciTools\sample\Zlib\zlib.udb C:\Users\Yvonne\AppData\Roaming\SciTools\sample\fastgrep.tdb C:\Users\Yvonne\AppData\Roaming\SciTools\sample\fdas\fdas.udb New Project Open Project Getting Started Get New Version! What's New In Understand 2.5? Understand Help Understand FAQ	<ul> <li>Forum Grand Opening (again) - Tue, 1</li> <li>Build 515 available - Mon, 10 May 201</li> <li>Build 513 ready for download - Mon, 7</li> <li>Build 512 Available - Fri, 09 Apr 2010</li> <li>Code Refactoring - Tue, 06 Apr 2010</li> <li>Build 511 Available - Fri, 02 Apr 2010</li> <li>Line and Statement Counting Metrics -</li> <li>Understand 2.5 Build 510 Available - S</li> <li>2.5 Upgrade Free if Maintenance Curre</li> <li>Awesome New Graph - Thu, 11 Mar 2</li> <li>Understand powers metrics in recent</li> <li>Support Forum temporarily disabled - </li> <li>Understand 2.5 - Tue, 09 Mar 2010</li> <li>Understand on Facebook - Become and an analysis</li> </ul>
Understand Perl API Help SciTools Support Licensing Open Sample Project	Visit ScīTools Blog Visit ScīTools Website Sign Up For Build Notices

	You can also choose File > Open > Project and File > Recent Projects from the menus to open projects.
	If you are learning about <i>Understand</i> , use the links in the <b>Getting Started</b> box. You can click <b>Open Sample Project</b> and choose an example project that uses a source code language used in your own projects.
	If you are a more experienced <i>Understand</i> user, use the links in the <b>News &amp; Announcements</b> box to keep your knowledge current.
	If you have closed the Getting Started tab and want to reopen it, choose <b>Help &gt; Getting</b> <b>Started</b> from the menus. If you don't want to see the Getting Started tab every time you run <i>Understand</i> , uncheck the <b>Show on Startup</b> box.
	When you are finished using a project, you can open another project or choose <b>File &gt; Close </b> <pre>project_name&gt;.udb</pre> . You will be asked if you are sure you want to close the project. If you have made any changes to files, you will be prompted to save or discard the changes for each file individually.
	If you want to make sure you have installed the latest version of <i>Understand</i> , you can choose <b>Help &gt; Check for Updates</b> from the menus. (You'll see the <b>Get New Version</b> button in the Getting Started tab if a new version is available.)
Other Ways to Run Understand	For information on running <i>Understand</i> from the command line, see Chapter 14, Command Line Processing.
	If multiple users will run <i>Understand</i> from the same Windows machine, each user may have a separate initialization file. These files store user preferences. <i>Understand</i> looks for the initialization file location in the following locations, depending on the operating system (on Windows, this location is referenced by the WINDIR environment variable):
	<ul> <li>Windows 2000/XP: C:\Documents and Settings\USERID\Application Data\SciTools\Understand.ini</li> </ul>
	Windows Vista/7/8: C:\Users\USERID\AppData\Roaming\SciTools\Understand.ini
	Linux/Unix: ~/.config/SciTools/Understand.conf
	Mac OS X: ~/Library/Preferences/com.scitools.Understand.plist

# **Context Menus Are Everywhere**

Right-clicking gets you a long way in *Understand*; almost everywhere you point, you can learn more and do more by bringing up menus with your right mouse button.

For example, if you	/* write a four	r-by	yte little-endian	unsigned i
right-click on an entity	local void put4	(ur	nsigned long val,	FILE *out)
you see the list of	{			
commands shown	putc(val &		View Information	
here.	putc((val >		Graphical Views	
Hold down the Ctrl key	putc((val >		Interactive Reports	
while right-clicking to	<pre>putc((val &gt; }</pre>		Edit Definition	
rather than re-using			Add to Favorites	· · ·
existing ones.	/* Load up zlik		Add Selection to Favorites	• • •
Remember to right-	local void zpul		Add Location to Favorites	+
click, anytime,	` if (in->lei		Annotate	
anywnere, on any	bload(i		Remove Annotations	
information about that	if (in->lei		User Tools	•
entity.	bail <b>("</b> ı		Explore	
	strm->avail		Engline	
	strm->next_			
	}		Metrics Charts	•
	/t Write header		Browse Metrics	
	local void gzir		Cut	Ctrl+X
	{		Сору	Ctrl+C
	<pre> fwrite("\x1 </pre>		Paste	Ctrl+V
	*crc = crc3		Select All	Ctrl+A
	*tot = 0;		Hide Inactive Lines	Ctrl+Alt+I
	}		Fold All	Ctrl+Shift+H
	/* Copy the cor		Soft Wrap	Ctrl+Alt+W
	block if cl:		Comment Selection	Ctrl+.
	boundary.		Uncomment Selection	Ctrl+Shift+
	the output,		Sart Calaction	
	crc and leng		Sort Selection	
	gzip file is		Change Case	<b>`</b> [
	or gzcopy()		Reindent Selection	Ē
	iocai void gzed		Reindent File	10
	{		Add Bookmark	
	<pre>int ret;</pre>		Previous Bookmark	
	int pos;	_	/* wher	e the "las

Show:	C++	Functions	▼					
Filter:			▼ Inf	armation Brows	ser			QAX
Show: Filter: adler3 adler3 allocat allocat allocat allocat allocat assert atol bail base bclose Begin( bget4 bi_flus bi_rev bi_wir bits bits		Functions         View Information         Graphical Views         Interactive Reports         Edit Definition         Edit Declaration         Add to Favorites         Annotate         Remove Annotations         User Tools         Explore         Find In         Metrics Charts         Browse Metrics	<ul> <li>✓ Info</li> <li>✓ Info</li> <li>✓ Info</li> <li>✓ Static</li> <li>Def</li> <li>Ret</li> <li>4 Par</li> <li>u</li> <li>F</li> <li>4 Cal</li> <li>P</li> <li>A ref</li> <li>D</li> <li>C</li> <li>C</li> <li>Met</li> <li>A ref</li> </ul>	Function Brows Function puts ined in: gzjoir urn Type: voi ameters m nsigned long v LE *out Is m utc led By m zcopy erences by F lefine gzjoin.c all gzcopy gzj all gzcopy gzj trics m thitectures	ser	View Information Graphical Views Interactive Reports Edit Definition Add to Favorites Annotate Remove Annotation User Tools Explore Find In Metrics Charts	le Sync	
bits blast blen		Browse Metrics Filter By Selection				Browse Metrics Show Linkname Expand Expand All Collapse Collapse All Copy Copy All	Right Ctrl+Shift+Right Left Ctrl+Shift+Left Ctrl+C Ctrl+A, Ctrl+C	

Right-clicking on an entity in the filter area and the Information Browser provides the following lists of options:

### **Quickly Find Things in Your Source**

*Understand* provides several ways to quickly locate items of interest in your source code. These features include the Filter Area, the Entity Locator, and the Find in Files dialog.

.....

Entity Filter

The filter area of the *Understand* window helps you quickly find things in your code by separating that database into lists of Files, Classes, Functions, Objects, Types, Macros, Subprograms, Packages, Modules, Blocks, Methods, Interfaces, SQL Tables, and more. The types of filters available depend on the languages you have configured your *Understand* project to understand.

After clicking in the filter area, you can type a letter to move to the first entity beginning with that letter in the current list.

By default, the *Information Browser* shows all known information about the selected entity. It is a key to navigating in *Understand*.



For details, see Entity Filter on page 129 and Information Browser on page 131.

Entity Locator

#### .....

The filter provides a quick way to find major items that were declared and used in your project. However, some items such as local parameters, variables, undefined (never declared or defined), and unresolved variables (declared but not defined) are not listed in the filters. To search or browse the entire database for your project, use the *Entity Locator*.

To open the *Entity Locator*, choose **View > Entity Locator**.

▼ Loc	ator: All Entities	s (8 of 4865 entitie	s)		9 t	۶×
Show:	All Entities				- 🔾 💭 🗆 Sync	÷
Entity	≜ E	Kind 🛔	Declared In 🛔	File 🛔	Date Modified	*
done	×				▼ 3/6/2012 4:56:29 PM ▼	j
done		Local Object	do_flush	gzio.c	Friday, October 23, 2009 11:50:32 AM	-
finish_	done	Enumerator	[unnamed]	deflate.c	Friday, October 23, 2009 11:50:32 AM	
gz_hea	ader_s::done	Public Object	gz_header_s	zlib.h	Friday, October 23, 2009 11:50:34 AM	Ξ
lOrigDo	ne	Local Object	main	testzlib.c	Friday, October 23, 2009 11:50:18 AM	
lOrigDo	ne	Local Object	main	testzlib.c	Friday, October 23, 2009 11:50:18 AM	+

By default, this area lists all the entities in the project. You can search for entities matching a particular text or regex string using the fields above each column.

For details, see Entity Locator on page 155.

As in any other window, the context menu is also active.

You can select multiple rows and columns and copy their contents to the clipboard. When you paste, the contents will be pasted as tab-separated text.

### Instant Search Instant Search lets you search your

Instant Search lets you search your entire project instantly, even if it contains millions of lines of source code. As you type, you can see terms that match the string you have typed so far.

A number of powerful search options are supported with Instant Search. See *Instant Search* on page 148.

	_ 8 ×
tes	× <u>?</u>
test	A
test_compress	
test_deflate	
test_dict_deflate	A
test_dict_inflate	
test_flush	
test_gzio	<b>T</b>

Find in Files Similar to the Unix co you may search files occurrence of a string Files either from the from a context menu.	Similar to the Unix command grep, you may search files for the	▼ Find In Files
	occurrence of a string. Select <b>Find in</b> <b>Files</b> either from the <b>Search</b> menu or from a context menu.	Find:  error    File Types:  **
	When you click <b>Find</b> , a list of all occurrences matching the specified string or regular expression is displayed in the <i>Find Results</i> window. Double click on any result to display the <i>Source View</i> where the string occurs. The options let you set behaviors such as case-sensitivity and wildcard pattern matching. See <i>Find in Files</i> on page 150 for more information.	Case Sensitive       Match Whole Words         Search Type:       Fixed String         Find In:       Project Files         Semantic Options       Semantic Options         Only Show Results In:       Comments         Strings       Statements         Inactive Code       Inactive Code

#### Favorites

You can place entities and code locations that you often use on your Favorites list. To add a favorite, right-click on it and select **Add to Favorites** along with the name of the list to contain this item.

To see the Favorites list, choose **View > Favorites** and the name of the list to open.



See Favorites on page 142 for more information.

### **Information Browser**

Just about everything *Understand* knows about code is shown in the Information Browser (IB). The IB is used for all types of entities.

The Information Browser shows different things depending on the type of entity selected.

It shows different kinds of information about entities such as source files, classes, members, functions, types, methods, packages, interfaces, and more. Information that is hierarchical in nature (such as a call relationship) can be expanded multiple levels.

Below are Information Browser windows for a file and a C function:



For details, see Information Browser on page 131.

### **Source Editor**

*Understand* has a source editor that not only lets you edit your source code, it colorizes the source code and tells you about the code you are editing.

Source can be visited by double-clicking almost anywhere else in the tool. You can move forward or backward through such "visits" by using the **Next** and **Previous** icons in the toolbar.



As with any other place in *Understand*, a context menu is available throughout the editor. To learn about something just right-click on it to see what information is available.

For details, see Source Editor on page 166.

### **Architecture Browser**

The Architecture Browser allows you to manage *architectures*. It shows a list of all the defined architectures in the database and provides a way to navigate individual architectures.

For example, this window shows the auto-architectures provided with *Understand*: Calendar, Directory Structure, Languages. The architectures are expanded somewhat here to show the top-level nodes for an example application.



You can use the auto-architectures, create your own architectures, import and export architectures (as XML files), generate graphs and metrics for any level in an architecture hierarchy, and combine architectures through filtering.

For details, see About Architectures on page 192.

### **Graphical Views**

Understand analyzes your software code and creates a database containing information about the entities and the relations between entities. The database can then be browsed using various "graphical view" windows. The graphical views are divided into these kinds:

- **Hierarchy** views show relations between entities. Each view follows a relation (for instance "Calls") from the starting entity (that you inquired about) through its children and successors.
- **Structure** views quickly show the structure of any entity that adds to the structure of your software (for instance a package, function, procedure, or task).

Examples of each type are shown in the following figure:



For details, see Using Graphical Views on page 245.

### **ASCII and HTML Reports**

Views in *Understand* provide information about individual entities. The reports bundle information about all entities in ASCII or HTML format.



The HTML and ASCII reports also show information not available interactively, such as project metrics and quality reports. These reports are suitable for printing or browsing with a web browser.

See Using Reports on page 207 for more information.

### **APIs for Custom Reporting**

*Understand* data is also available directly from scripts and programs that you (or we) write. A C API (usable from C, C++ or other languages that can call C libraries), a Python interface, a Java interface, and a Perl interface are provided with *Understand*.

Using the API, you have exactly the same access that we have when we write the existing GUI and report generators.

This manual doesn't cover the APIs. Choose **Help > PERL API Documentation** or **Help > Python API Documentation** for more information. Java API documentation is provided in the doc/manuals/java subdirectory of the *Understand* installation. Tutorials for the APIs are available online at scitools.com/api-tutorials/.

The **Reports > Project Interactive Reports** and **Graphs > Project Graphs** commands display a list of user-created plugins, which can be created using the Perl API. For information about creating plugins, please contact support@scitools.com. The SciTools Support website at www.scitools.com/support and the SciTools blog at scitools.com/blog also contain information concerning plugins.

# Chapter 3 Configuring Your Project

This chapter shows how to create new *Understand* project files that you will use to analyze your source code.

This chapter contains the following sections:

Section	Page
About Understand Projects	34
Creating a New Project	35
Project Configuration Dialog	39
Languages Category	41
Files Category	42
File Types	48
File Options	49
Scheduled Activities	50
Metrics	52
Reports	54
Visual Studio	57
Annotations	59
Ada Options	61
Assembly Options	65
COBOL Options	66
C++ (Fuzzy) Options	68
C++ (Strict) Options	75
C# Options	79
Fortran Options	81
Java Options	84
JOVIAL Options	86
Pascal Options	88
PL/M Options	90
Python Options	92
VHDL Options	93
Web Options	93
Setting General Preferences	95
Analyzing the Code	121

### **About Understand Projects**

Understand is like a compiler, except it creates information, not executable code.

In order for *Understand* to analyze your source code, it needs much of the information your compiler needs. It needs to know:

- What source files to analyze
- The type of source code
- · The standard library paths and include directories
- Where to find Java .jar files that provide classes for which you do not have source code
- Compiler/environment specific macros that need to be defined for the pre-processor
- Application-specific macro definitions
- What implementation parameters (such as integer precision) and column truncation settings to use
- Any namespaces

If you developed the program or have been working with it for some time, this information is probably obvious to you. However, if you inherited this source code from another programmer, team, or company, you will probably have to examine the project building files (for example, a makefile) in order to come up with the information needed for accurate analysis of the code.

The easiest way to analyze your code is to use *Understand*'s GUI to build and analyze a project. This chapter will walk you through that process.

.....

The UnderstandThe Understand project database is stored in a proprietary binary format. The fileProject Databaseformat uses a network/object format that is optimized for storing Understand<br/>information.

Understand databases have a file extension of \*.udb.

The project database permits multiple simultaneous read accesses, but it does not support multi-user write access. You will see a message if the project database is locked.

Occasionally, a new feature to *Understand* requires a change to the database format. Such changes are noted in the Change Log. When you install a build that modifies the database format, existing projects are automatically re-analyzed when you open them.

## **Creating a New Project**

	To begin analyzing code, you create a project and specify what source files to analyze. <i>Understand</i> analyzes your code and creates a database you can browse. This database can be refreshed incrementally in the GUI or updated using command-line tools.
	This section shows how to create a new project. The project will be stored in a <i>Project Database</i> , which has a file extension of .udb.
	To create a new project, follow these steps:
	<ol> <li>Click the New Project link in the Getting Started tab that you see when you start Understand. Or, choose File &gt; New &gt; Project from the menus.</li> </ol>
	- By default, this opens the New Project Wizard, which is described on page 35.
	<ul> <li>Alternately, you may have disabled the option to run this wizard, in which case, you see the "Create new project as" dialog. Browse to the folder where you wish to create the project database. Type the name of the project in the File name field. A *.udb file extension will be added automatically. Click Save. You will see the Understand Project Configuration dialog, which is described in page 39.</li> </ul>
	<ul> <li>Another way to create a project is to add Buildspy to your gcc/g++ build process. This automatically generates an Understand project when you compile your project. See Using Buildspy to Build Understand Projects on page 329.</li> </ul>
New Project Wizard	Unless you have disabled the New Project Wizard, this is the tool you use to create projects. To open it, click the <b>New Project</b> link in the Getting Started tab that you see when you start <i>Understand</i> . Or, choose <b>File &gt; New &gt; Project</b> from the menus.
	1 In the Create a Project File page of the wizard, type a <b>Name</b> for the project and browse for a directory to contain the <i>Understand</i> project files. It is often handy to have the project file in the top-level directory of the source code, but this is not required. If the directory does not exist, you are asked if you want it created.
🔎 New P	roject Wizard
Creat	te a project file



2 Click **Next** to see the Languages page of the wizard. The defaults for this and other configuration categories are the most recent settings you saved for another project.

Languages Select the source code language(s) that your project will contain. Later, you can configure options for how each language you select is handled. C/C++ Fuzzy Analysis: Great for the first pass at most code, since very little setup is required. Uses fuzzy logic to handle incomplete, non-compiling code gracefully and as accurately as possible. C/C++ Strict Analysis: This option may result in a more accurate analysis, so more setup is required. Include paths and macros will need to be defined during the analysis. Handles C++ templates and overloaded functions better than the fuzzy analyzer. It also will analyze Objective C/C++.	<ul> <li>Ada</li> <li>COBOL</li> <li>Coldfire 68K Assembly</li> <li>C/C++ Fuzzy Strict*</li> <li>C#</li> <li>Fortran</li> <li>Java</li> <li>Jovial</li> <li>Pascal</li> <li>PL/M</li> <li>Python</li> <li>VHDL</li> <li>Web**</li> <li>Import project settings from a MS Visual Studio Project (C/C++/</li> <li>* Strict includes support for Objective-C and Objective-C++</li> </ul>
overloaded functions better than the fuzzy analyzer. It also will analyze Objective C/C++.	Web**  Import project settings from a MS Visual Studio Project (C/C+  Strict includes support for Objective-C and Objective-C++  Web languages include CSS, HTML, Javascript, PHP, & XML

- **3** Put checkmarks next to languages used in the source code for this project. See *Languages Category* on page 41 for more information about specific languages and the strict C/C++ option.
- 4 If you use Microsoft Visual C for your C, C++, or C# code, you can check the box to import project settings. Then click **Next**.
5 If you checked the **Import project settings from a MSVC Project** box, you see the Visual Studio File(s) page. Otherwise, skip to the next step.

Visual Studio File(s)	Visual Studio File Configuration Exclude Filter	Add
Optional If you use Visual Studio, you can synchronize your Understand project with Visual Studio projects by adding them in this dialog. Understand uses the source files, includes, macros, and other settings from your Visual Studio projects. As your Visual Studio projects change, Understand also updates its project settings.	Unfiltered Contents	Edit

To synchronize your Understand project with Visual Studio projects, click **Add**. In the Add a new Visual Studio file dialog, click ... and browse for your Visual Studio project file. In the Add a new Visual Studio file dialog, select the project configuration you want used when *Understand* analyzes your project. Then click **OK**. See *Visual Studio Studio* on page 57 for more information.

You can add multiple Visual Studio projects or use the **Edit** button to change the Configuration setting. Then click **Next**.

6 In the Source Files page of the wizard, you add source files to a project by clicking Add a Directory or Add a File.

Source Files Add directories that contain the source files you want analyzed. You may choose whether or not to automatically include subdirectories. Files that match languages you selected are added to the project. You do not need to add files included from other libraries, since those can be added later when you identify include directories	0 <u>F</u> iles 🚞 Add a <u>D</u> irectory 📄 Add a <u>F</u> ile ▼	🖌 Remove 💽	Portability
Cancel	[	< Back	Next >

The **Portability** button lets you choose whether file paths will be stored using absolute paths, relative paths, or user-defined root path variables. See page 47.

To add a file, just browse for the file and add it.

When you add a directory, you can browse for a directory, modify the list of languages used in the source files, add additional filters for file extensions not expected by *Understand*, filter out any files you want to exclude (for example, temp\*.\*), and choose whether all the subdirectories of this directory should be added. You can also choose whether the directory will be watched for changes. See *Adding Directories* on page 43 for details.

🔎 Add a Project 🛛	Directory	
Directory	C:\software\queue	
Configured Filters	C, C++, Java, VHDL (*.c;*.C;*.H;*.cc;*.cpp;*.cxx;*.h;*.hh;*.hpp;*.hx 🔻 🛄	
Additional Filters	▼	
Exclude	* · · · · · · · · · · · · · · · · · · ·	
Include subdire	ctories	
Watch this directory The directory will be watched for any on disk changes. If a file is deleted from disk, it is removed from the project. If new sources are added to this directory or subdirectory of sources is added, it is automically added to the project. Those files and directories must match above filters and settings.		

If you already chose a Visual Studio project, those files are automatically listed in the Source Files page of the New Project Wizard.

If you want to delete a file or a directory (and its subdirectories), select that item and click the "X" icon. Click **Next** when you have finished selecting files and directories.

7 Choose whether to Analyze project now or further configure the project. Choosing Configure more settings takes you to the Project Configuration dialog, which is described starting on page 39. In either case, you can go to the Project Configuration dialog anytime you like.

P New Project Wizard (C:\temp\MyUnderstandProject.udb)			
Ready to analyze You've finished providing the settings needed to create an Understand project. You can choose to configure more settings (such as language-	<ul> <li>Analyze project now</li> <li>Begin analyzing project sources. During the analysis you may be prompted</li> <li>for optional configuration items. They may add further information to your parsed results but are not required and can be safely ignored.</li> </ul>		
specific settings) using the Project Configuration dialog. Or you can choose to analyze the project now and start using Understand to explore your code. In either case, you can go back to the Project Configuration dialog anytime you like.	Configure more settings Present the Project Configuration Dialog to view language specific Configurations. All required configurations have been initialized to most common defaults. This can be invoked later at any time from the Project menu Use the new project wizard when creating new projects		

## **Project Configuration Dialog**

The Understand Project Configuration dialog opens when you create a new project or when you choose the **Project > Configure Project** menu item.

The categories on the left in the Project Configuration dialog allow you to specify various project settings to be used during analysis. The Project Configuration dialog contains the following categories:

- Languages: Set the types of languages to be analyzed. For details, see *page 41*.
- **Files:** Set the locations of source files to be analyzed. For details, see *page 42*.
- File Types: Set how to handle source file types and what file extensions are used. For details, see page 48.
- File Options: Set the file encoding and editing mode for source files. For details, see *page 49*.
- **Scheduled Activities:** Schedule events to take place at regular intervals. For details, see *page 50*.
- **Metrics:** Select the metrics you want computed for this project. For details, see *page 52*.
- Understand Project Cor Languages Files File Types File Options Scheduled Activities Metrics Reports Visual Studio Annotations Ada Assembly COBOL C++ b C# Fortran Java Jovial Pascal PL/M Python Web
- Reports: Select reports you want generated. For details, see page 54.
- **Visual Studio:** Select a Visual Studio project to synchronize this *Understand* project with. For details, see *page 57*.
- Annotations: Set how to store and display annotations. For details, see page 59.
- Language-Specific Options: Set options for the languages you selected in the Languages category. For details, see:
  - Ada Options, page 61
  - Assembly Options, page 65
  - COBOL Options, page 66
  - C++ Options, page 68
  - C++ Strict Options, page 75
  - C# Options, page 79
  - Fortran Options, page 81
  - Java Options, page 84
  - JOVIAL Options, page 86
  - Pascal Options, page 88
  - PL/M Options, page 90
  - Python Options, page 92
  - Web Options, page 93

For advice about ways to adjust the project configuration to improve the accuracy of project analysis, see the SciTools website.

After you change the project configuration, click the **OK** button and the configuration will be saved. Whenever you modify the files in the project configuration, including at the time of project creation, a dialog alerting you to the change in configuration appears.



Click **OK** and *Understand* begins analyzing (also called parsing) the code (page 121).

If you want to close the Project Configuration dialog without saving any changes, click **Cancel**, and then click **Yes** in the box that asks if you really want to cancel changes.

If you want to make a copy of the current configuration, for example to create two variants of one configuration, choose **File > Save** *project.udb* **As**. This makes a copy of the \*.udb file and modifies uses of the database name within the configuration as needed.

#### Languages Category

In the **Languages** category of the Project Configuration dialog, you can check boxes for the languages used in your project. A project can contain source code in one or more languages.

Understand Project Co	nfiguration: C:\temp\MyUnderstandProject.udb
Languages         Files         File Types         File Options         Scheduled Activities         Metrics         Reports         Visual Studio         Annotations         Ada         Assembly         COBOL         C++         C#         Fortran         Java         Jovial         Pascal         PL/M         Python         Web	<ul> <li>Ada</li> <li>COBOL</li> <li>Coldfire 68K Assembly</li> <li>C/C++</li> <li>Fuzzy Strict*</li> <li>C#</li> <li>Fortran</li> <li>Java</li> <li>Jovial</li> <li>Pascal</li> <li>PL/M</li> <li>Python</li> <li>VHDL</li> <li>Web**</li> <li>* Strict includes support for Objective-C and Objective-C++</li> <li>** Web languages include CSS, HTML, Javascript, PHP, &amp; XML</li> </ul>
	OK Cancel

When you select a language, categories for that language are added to the list on the left in the Project Configuration dialog. The languages you choose here not only affect how the source files are analyzed. They also affect the filter types available, the metrics available, and the report types available.

If you select multiple languages, references between those languages are analyzed. For example, if C code calls a Java function, that reference will be found.

If you have C or C++ code, you can decide to use either the default C/C++ analyzer (the "fuzzy" analyzer) or the newer "strict analyzer". To use the strict analyzer, check the **Strict** option next to C/C++. Internally, checking this box causes a completely separate analyzer to be used to analyze your C/C++ code.

The "strict analyzer" provides the following features:

- Support for Objective-C and Objective-C++ (used for Mac OS and iOS) is provided with the strict analyzer, but not with the default C/C++ analyzer.
- Provides better support for Templates.
- Provides better support for Overloaded functions.

The default C/C++ analyzer aims to use fuzzy logic to handle incomplete, noncompiling code gracefully and as accurately as possible. The new analyzer is more strict than the old analyzer and requires a more accurate project definition—for example, by specifying all include paths and macro definitions and including only those files in the project that are used in the software build. For details, see the Creating Accurate C/C++ Projects web page.

If you are using the strict analyzer, see C++ (*Strict*) Options on page 75 for how to configure your project.

#### **Files Category**

In the **Files** category of the Project Configuration dialog, you can add source code directories and/or individual files to the project. You can also delete specific files from the analysis and modify language-specific options for individual directories and files.



You can add source files here, or you can tie the project to those specified in an MS Visual Studio project file (MS Windows versions of *Understand* only). See *Visual Studio* on page 57.

The top area shows the directories and files you have added in a tree that you can expand. It also shows how many files are currently in the project.

The bottom area shows any overrides you have set for the selected directory or file.

Icons at the top of the dialog perform the following actions:

- Open the Add a Directory dialog.
- Open the Add a File dialog.
- Choose Add a File or import a list of files
- > Delete the selected directory or file from the project analysis.
- Copy the override settings for the selected directory or file.
- Paste the override settings to the selected directory or file.
- Configure override settings for the selected directory or file.

Note that your changes are not saved until you click OK.

Click **Portability** to set portability options for file paths. See page 47.

Click **Rescan** if you have added files to a directory that are not shown in this dialog.

#### **Adding Directories**

To add source directories to the project, click 🖰. You see the Add a Directory dialog:

Add a Project Directory		
Directory	C:\myCode\includes	
Configured Filters	Assembly, C, C++, CSS, Html, Java, Javascript, Php, Python, Xml ( 👻 🛄	
Additional Filters	<b></b>	
Exclude	*	
<ul> <li>Include subdirectories</li> <li>Watch this directory</li> <li>The directory will be watched for any on disk changes. If a file is deleted from disk, it is removed from the project. If new sources are added to this directory or subdirectory of sources is added, it is automically added to the project. Those files and directories must match above filters and settings.</li> </ul>		

- 1 In the **Directory** field, type the full directory path. Or, you can click the ... button and use the Browse for Folder dialog to locate a directory containing source files and click **OK**.
- 2 In the Configured Filters field, click the ... button if you want to add or delete languages from the list shown. In the Select Filters from Configured File Types dialog, put a checkmark next to any languages you want to be recognized as part of the project. Notice that additional languages are listed beyond those shown in the

Languages category. These include Basic, MSDos Batch, Perl, Tcl, Text, and Verilog.

If this directory contains source files with extensions that are not listed, click **Configure**. Also, see *File Types* on page 48. For example, you might add \*.a64 as an assembly file type.

- 3 In the Additional Filters field, type a pattern-matching string that matches only the files you want to keep in the analysis. For example, std\*.\* includes only files that begin with "std". You can separate filters with a comma.
- 4 In the Exclude field, type a pattern-matching string that matches files you want to exclude from the analysis. For example, temp\*.\* excludes all files that begin with "temp". You can separate filters with a comma.
- **5** To select and add multiple subdirectories to a project configuration, check the **Include subdirectories** box (on by default). This causes all source files matching the filter in all subdirectories of the specified path to be added to the project.
- 6 If you want this directory to be watched for any new files or deleted files, check the Watch this directory box. Whenever a source file is added to or deleted from this directory, the change is reflected in this project. Watched directories are indicated by the icon in the files list. Directories excluded from being watched are indicated by the icon. By default, the subdirectories of a watched directory are also watched. See page 45 for watch setting overrides.
- 7 After you have set the fields, click the OK button to add the source files in that directory to the project. You can click Cancel if the add file process is taking too long.
- *Tip:* You may add files from multiple directory trees.

If you are using Microsoft Windows, you may drag and drop a directory, a file, or a selection of files, from another window into the Project Configuration dialog to add it to the project. If you drag a folder, the Add a Project Directory dialog opens automatically. If you drag an individual file, that file will be added to the project whether it matches the file filter or not.

All directory paths are absolute.

To add individual source files to the project, click 
Source files a file selection dialog, Adding Files which allows you to select one or more source files to add to the project. Browse for and select a file or files. Then click **Open**. The file(s) are added to the project. If you click the - down arrow next to the r icon, you can choose to import a text file that contains a list of source files to import. For example, you might generate such a file from a compiler application or code management system. The file should contain one absolute file path per line. See Adding Files to a Project on page 322 for an example of such a file. Removing To remove a directory or file from the project, select the items you want to remove and **Directories and Files** click  $\mathbf{X}$ . The directory or file itself is not deleted from the file system. You can right-click on a removed file or directory and choose Add file to project or Add directory to project to re-add it to the project.

## **Setting Overrides** Normally, each file in the project is processed according to the rules you specify in the Project Configuration window for the language of the file. For example, for C++ you can

Project Configuration window for the language of the file. For example, for C++ you can set include directories and macro definitions. However, you can override the default settings on a directory-by-directory or file-by-file basis if you like.

Directory: To override settings for a directory, follow these steps:

- 1 Select a directory.
- 2 Click **b** or right-click and select **Configure override settings**.

Tirectory: C:\Program Files\SciTools\sample\fastgrep			×
Watched Properties         ▲ Override         File Types         File Encoding         Ada Macros         C++ Includes         C++ Auto Includes         C++ Macros         C# References         Fortran Includes         Fortran Macros         Java Linkage         Pascal Includes         VHDL Options	Watch Settings Configured Filters Additional Filters Exclude Include subdire	Watch this directory  Assembly, C, C++ (*.s;*.c;*.C;*.H;*.cc;*.cpp;*.cxx;*.h;*.hh	
Use Defaults OK Cancel			

- 3 In the **Watched Properties** category, you can choose how files in this directory should be watched for new files to add to the project or deleted files to remove from the project. For **Watch Settings**, you can choose to watch a directory, not watch a directory, or inherit watch settings from the parent directory. In addition to specifying whether to watch a directory, you can set filters and exclude filters for an individual directory that control what types of new and deleted files will be found.
- 4 In the various **Override** categories, you can make directory-specific languagerelated settings. The list of categories depends upon the languages enabled in your project.

The **File Type** category lets you override the language of this file indicated by the file extension. The **File Encoding** category lets you override the encoding setting described in *File Options* on page 49.

File: To override settings for a file, follow these steps:

- 1 Select a file.
- 2 Click or right-click and select Configure override settings.

	File: C:\Program Files\SciTools\sample\fastgrep\egrep.c
	Watched Properties Override File Type File Encoding C++ Includes C++ Macros Language: Project default (C)
	Use Defaults OK Cancel
	3 In the various <b>Override</b> categories, select a category and make changes. The categories available are different depending on the language of the source file. See page 61 through page 90 for details. The <b>Watched Properties</b> category is available for file overrides if you are using Relative or Named Root portability.
	4 Click <b>OK</b> to save your overrides.
	Special icons in the directory tree indicate which directories are being watched $\overline{\mbox{cons}}$ , have overrides $\mbox{cons}$ , or both $\mbox{cons}$ .
	The various <b>Override</b> categories have an <b>Ignore Parent Overrides</b> checkbox. Checking this box makes only the override settings you apply at this level (directory or file) apply; settings from higher levels are not inherited.
Scanning Watched Directories	If you set directories to be watched, you can scan those directories for new files to be added or deleted files to be removed by choosing <b>Project &gt; Rescan Project Directories</b> .
If files are found that you don't want to include in the project, uncheck the boxe those files to exclude them from the project configuration. Directories that you include in the project are automatically scanned for new fi you use <b>Analyze All Files</b> to analyze the project.	

#### Setting File Portability

You can control the portability of *Understand* projects by clicking the **Portability** button at the top of the **Files** page of the Project Configuration dialog. You will see the following dialog.

Project File Portabil	lity	? X
Portability Options		
Select a project porta stored as absoute pa user-defined root pat	ability mode that best suits your needs. "Absolute" mode is the least portable, as all file pa aths. "Relative" mode stores file paths relative to the project directory. "Named Roots" mo th variables.	iths are ide uses
File Portability Mode:	Absolute	-
Use File Portability	y Mode to convert paths of all files and directories in the project	
Edit Named Roots	ОК	Cancel

A more portable project can allow you to share the project with other users and to use the project unchanged after moving the source code files.

The choices are as follows:

- **Absolute:** This option is the default. It stores full file paths for all directories. If the source files change location, the paths will be incorrect.
- **Relative:** This option stores the relative path to directories from the location of the *Understand* project database. If you store the project database in the source file tree and move it along with the source files, the project can still be used.
- **Named Root:** This option allows you to specify "Named Roots" that are similar to environment variables to point to a root directory. Different users may then use different definitions for a named root. Click the **Edit Named Roots** button and see page 107 for details.

Check the **Use File Portability Mode to convert paths** box if you want all the file paths currently stored in the project to be updated when you click **OK**.

### **File Types**

In the **File Types** category of the Project Configuration dialog, you can control how file extensions are interpreted by *Understand*.

Understand Project Co	nfiguration:	C:\temp\MyUnder	standProject.ud	b 💌
Languages Files File Types File Options Scheduled Activities Metrics Reports Visual Studio Annotations C++ C# Java Web	Extension .C .H .TXT .a .ada .ada .adb .ads .bas .bas .bat .c .cbl .cc .cgi .cob .cpl .cpp .cpy	Language C++ C++ Text Ada Ada Ada Ada Basic MSDos Batch C COBOL C++ Perl COBOL Jovial C++ COBOL		New Edit Delete

The list shows all the file extensions already understood. Files with the types understood for the languages you checked in the Languages category are analyzed as part of the project. Other file types are not analyzed.

To modify an existing type, select the type and click **Edit**.

To add a file extension to the list, click **New**. Type a file extension and select the language to use for the file extension. Then click **OK**.

🔎 Add a New File Type	x
Extension:	
Language: C++	-
OK Cancel	

The file extension you type should begin with the period. It can contain simple \* and ? wildcards.

Certain file types may be interpreted differently depending on the languages you selected. For example, in a Visual Fortran project, .h files are interpreted as Fortran files, rather than as C headers files.

## **File Options**

In the **File Options** category of the Project Configuration dialog, you can control how files are opened and saved by *Understand*.

Understand Project Co	nfiguration: C:\temp\MyUnderstandProject.udb
Languages Files File Types File Options Scheduled Activities > Metrics > Reports Visual Studio Annotations > C++ > C# > Java Web	Encoding File Encoding System Editing Options Open all project files as read only files File Size Maximum size for files to be analyzed: 8  MB
	OK Cancel

- File Encoding: Select the type of encoding to use when saving source files. Many encoding formats are supported. You should change this only if your other applications have problems opening or displaying files saved by *Understand*. See *Editor Category* on page 109 for more information. The default file encoding is "System", which means the default encoding for your computer. If you change the setting here, new projects you create use the last setting you saved. You can override the file encoding setting on a file-by-file or directory-by-directory basis (see *Setting Overrides* on page 45).
- **Open all project files as read only files:** Check this option if you do not want files to be edited and saved within *Understand*.
- **Maximum size for files to be analyzed:** Limits the size of files analyzed by *Understand*. You can use this option to exclude very large files. The default is 10 MB. An error message is provided if you attempt to edit a file that is too large to open.

## **Scheduled Activities**

In the **Scheduled Activities** category of the Project Configuration dialog, you can cause certain events to be performed on a regular basis. You can also open this dialog quickly by choosing **Tools > Scheduler > Scheduled Activities — <project\_name>**.

Understand Project Con	nfiguration: C:\temp\MyUnderstandProject.udb
Languages Files File Types File Options Scheduled Activities Metrics Reports Visual Studio Annotations C++ Options Includes Macros C# References Preprocessor Symbols Java Options Class Path Web	Process Project History  Process At: 12:07:23 PM  Mon Tue Wed Thu Fri Sat Sun  Process every: 0  Minute(s)  Do not process.  Rescan watched directories Analyze All Files Analyze Changed Files Metric Processing Metric CSV Export Metric CSV Export Metric CSV Export Settings: Output Directory: C:\tempMyUnderstandProject.csv Overwrite existing Metric CSV Export. Do not automatica Metric Export HTML Metric Export HTML Metric Export HTML Settings: Save output to: C:\temp Overwrite existing Metric Export HTML. Do not automatic
- III	Suppress Scheduler MessageBox

To schedule events for the project you currently have open, follow these steps:

- 1 Check the Process At box.
- 2 Select either a processing time, a processing interval, or to not process events. For a processing time, check the boxes for one or more days of the week. For a processing interval, specify a number of minutes up to 1440 (24 hours).
- 3 Check the boxes for the events you want performed. The events occur in the sequence shown. For example, watched directories are scanned before the project is analyzed, and the project is analyzed before metrics are processed.
- *Note:* Understand must be running at the processing time or the events will not occur.

The following activities are available for scheduling:

- Rescan watched directories: Check this box to automatically check for files that have been added to or deleted from project directories. See *Adding Directories* on page 43 for how to specify which directories to watch. If you have watched directories, you should always run this task before the "Analyze all files" task. To run this action without scheduling it, choose **Project > Rescan Project Directories**.
- Analyze all files: Check this box to automatically analyze all project files as described in *Analyzing the Code* on page 121. Run this task before generating any metrics so that the statistics will reflect the current state of the project. To run this action without scheduling it, choose **Project > Analyze All Files**.
- Analyze changed files: Check this box to automatically analyze any project files that have changed as described in *Analyzing the Code* on page 121. Run this task before generating metrics so that statistics will reflect the current state of the project. To run this action without scheduling it, choose **Project > Analyze Changed Files**.
- **Metric processing:** Check this box to automatically calculate project metrics. The metrics selected in *Metrics > Selected Category* on page 53 are processed. Run this task if you plan to schedule either of the following metrics export tasks.
- Metric CSV export: Check this box to automatically export metrics as a commaseparated value file. If this box is checked, you can select the directory path and output filename for the export. By default, any existing file with the same name is renamed to provide a backup. You can check the **Overwrite** box if you simply want to replace the old export file. To further configure the export, see *Metrics* on page 52. To run this action without scheduling it, choose **Metrics > Export Metrics** and see *Exporting Metrics to a CSV File* on page 235.
- Metric export HTML: Check this box to automatically export metrics as web pages. If this box is checked, you can select the directory path for the export. By default, any existing file with the same name is renamed to provide a backup. You can check the **Overwrite** box if you simply want to replace the old export file. To run this action without scheduling it, choose **Metrics > Project Reports** and see *Exporting Metrics* to *HTML* on page 234.

When scheduled activities are about to run, you see a dialog that gives you a chance to cancel the action. You can avoid this message by checking the **Suppress Scheduler MessageBox** in the configuration.

🔎 understand
Performing scheduled actions in 28 seconds
OK Cancel

If you schedule activities, you see a message that asks if you are sure you want to prevent the scheduled activities from running when you exit *Understand*.

To see a list of all projects for which you have scheduled activities, choose **Tools > Scheduler > Scheduled Activities — All Projects**. To change these times, you must open the project and then use the Project Configuration dialog for that project.

#### **Metrics**

In the **Metrics** category of the Project Configuration dialog, you can control how metrics are generated when a CSV file is exported. These options set the defaults for both manual updates (page 235) and scheduled automatic updates (page 50).

The Metrics category has two sub-categories: Options and Selected.

Duderstand Project Configuration: C:\temp\MyUnderstandProject.udb				
Languages Files File Types File Options Scheduled Activities Metrics Options Selected Reports Visual Studio Annotations C++ C# Java Web	Output File: C:\temp\MyUnderstandProject.csv			
	OK Cancel			

You see this window when you choose the **Project > Configure Project** menu item and then the **Metrics** category. If you attempt to generate metrics before configuring metrics, this window opens automatically.

The **Options** subcategory has the following fields:

- **Output file:** Specify the location and name of the file you want to use for metrics output. *Understand* sends its metrics output to a \*.csv (comma-separated values) file. This file can be opened with Microsoft Excel and other spreadsheets.
- Show File Entities Name as: Specify whether files should be displayed with Short names (just the filename), Full names (including the absolute path), or Relative names (relative directory path).
- Show Declared in File: Check this box if you want the file in which each entity is declared to be included in the output. You can specify whether you want these files displayed with Short names, Full names, or Relative names.
- Show Function Parameter Types: Check this box if you want the type of each function parameter listed.
- Write Column Titles: Check this box if you want column headings in the CSV file.

 Metrics > Selected
 The Selected subcategory has lists like the following:

 Category
 Category



- 1 In the **Available Metrics** list (left), select metrics you want to include in the output you generate. You can hold down Shift to select a continuous group or Ctrl to select discontinuous items.
- 2 Click Add to copy the selected metrics to the right column.
- **3** You can reorder the metrics in the right column using the **Move Up** and **Move Down** buttons.

The metrics available depend on the languages used in your project. See scitools.com/support/metrics\_list/ for descriptions.

#### Reports

In the **Reports** category of the Project Configuration dialog, you can control how reports are generated. The Reports category has the following sub-categories: **Output**, **Options** and **Selected**.

Languages Files	Generate Html
File Types File Options Scheduled Activities ▷ Metrics ■ Reports Output	Single       Alphabetic       Every n Entities       250         Title page:
Options Selected Visual Studio Annotations D C++ (Strict) Web	Save in <u>directory</u> : AppData\Roaming\SciTools\sample\fastgrep\fastgrep_html
	Single Text File Separate Files
	Save single text output file as: loaming\SciTools\sample\fastgrep\fastgrep.txt          Separate files directory:       aming\SciTools\sample\fastgrep\fastgrep_text          Clear directory before each report generation.
11	OK Cancel

This window opens if you choose the **Project > Configure Project** menu item and then the **Reports** category. You can also reach this window by clicking **Configure** in the Project Reports window.

You can control the colors and font styles in HTML reports as described in *Customizing Report Colors* on page 210.

Reports > Output Category	The <b>Output</b> subcategory has two main areas:		
	• <b>Generate HTML:</b> This option causes the report generation to create a large group of HTML files that are interlinked.		
	<ul> <li>You may generate Single or multiple HTML files for each report type. It is recommended that you split up the files for large projects. Choose Alphabetic to generate multiple HTML files per report that are split up alphabetically by the first letter of the entity name. Choose Every n Entities to generate multiple HTML files per report that are split up every "n" number of entities. By default, a single HTML file is generated for each letter of the alphabet.</li> </ul>		

- The "home" page for reports is index.html. You can select an alternate Title Page. - The default **Save in directory** is the *<proj file>* html folder below the folder where your \*.udb file is stored, but you can select an alternate location. - You can choose to clear the contents of the previously generated reports and anything else in the selected directory at the beginning of the report generation. Generate Text: This option causes the report generation to create simple text files containing the report data. - You may generate one text file of the specified location and name (by choosing **Single Text File**). Alternately, you may generate multiple text files (by choosing Separate Files) and specify a directory to contain all the files. The file extensions of each text file will denote the separate reports. Depending on which option you select, you can also select either a file or directory location for the output. You can choose to generate either or both of the HTML and text report formats. **Reports > Options** You can use the **Report > Options** category to control the contents and headers of Category reports. Languages Display full filenames Files File Types Write generation time on report File Options Scheduled Activities Display parameters Metrics Index by method name Reports Output Report Header Text Selected Left aligned text: Report generated by Understand Visual Studio Annotations Right aligned text: C++ (Strict)

The **Options** category has the following fields:

- **Display full filenames:** If you check this box, the invocation tree and metrics reports show full entity names. The default is to use short names.
- Write generation time on report: If you check this box, the generation date and time are included at the top of text report files. This is on by default.
- **Display parameters:** If you check this box, reports that list the names of functions and similar entities also include any list of parameters declared for that function.
- Index by method name: If you check this box, entities are sorted in the data dictionary, index, and reports by their short names, rather than full names (for example including the class path).
- Left aligned text: If you check this box, the text "Report generated by Understand" will be printed in the upper-left corner of each page of the text report.
- **Right aligned text:** If you check this box, the text you provide will be printed in the upper-right corner of each page of the text report. This text can be up to 45 characters.

#### Reports > Selected Category

The **Selected** subcategory lets you check the boxes for the reports you want to generate. The list of reports differs depending on which languages are used in your project. See Chapter 8 for descriptions of these report formats.



## Visual Studio

In the **Visual Studio** category of the Project Configuration dialog, you can tell *Understand* to use the source, macro, and include path settings from a Microsoft Studio project file.

You see this window when you choose the **Project > Configure Project** menu item and select the **Visual Studio** category.

Understand Project Co	onfiguration: C:\ten	np∖MyUnderstan	dProject.udb	×
Languages Files File Types File Options Scheduled Activities Metrics Reports Visual Studio Annotations C++ C++	Visual Studio File	Configuration	Exclude Filter	Add Edit Remove
⊳ C# ⊳ Java Web	Unfiltered Contents	S		
			ОК	Cancel

Follow these steps:

- 1 Click Add.
- 2 In the Add a new Visual Studio file dialog, click the "..." button next to Visual Studio File. Then browse to select a Visual Studio project file and click Open. MS Visual Studio project files with extensions of .csproj (C# project), .dsp, .dsw (workspace file), .sln, .vcp (Windows CE project), .vcproj (Visual C project), .vcxproj (VS2010 project), .vfproj (Visual Fortran project), and .vcw (workbench file) are supported.
- **3** Select the **Configuration** you want *Understand* to use when analyzing your project. You can select a project configuration or a solution configuration.
- 4 You can type an **Exclude Filter** to specify file extensions to exclude when importing a Visual Studio project.

**5** You can expand the **Unfiltered Contents** list to see the includes, defines, and files for the configuration currently selected.

🔎 Add a new Visual S	Studio file		
Visual Studio File:	C:\software\samples\sample.vcproj		
Configuration	DebuelWin32		
comgaration.	Debugiwinsz		
Exclude Filter:			
Unfiltered Contents			
▲ C:\software\sam	ples\sample.vcproj		
▲ Define			
UNICODE			
_UNICODE			
C:\software\samples\readme.txt			
C:\software\s	C:\software\samples\tarnetver.h		
▲ C:\software\s	amples\SomeProjName.cpp		
⊿ Define	⊿ Define		
WIN32			
_CONSOLE			
_DEBUG			
A C:\software\s	amples/stdatx.cpp		
WIN32			
CONSOLE			
_DEBUG			
	OK Cancel		

6 Click **OK** to add this to your project.

*Note:* If you sync with a Visual Studio workspace file, the default target is used because there is no mechanism for specifying targets for each .dsp project within a .dsw file.

Once set, the source files, macros and include paths from the Visual Studio project are used by *Understand*. This is in addition to any project settings you configure in the other categories.

*Note:* Settings in other categories for include path and macros take priority over the Visual Studio project settings. This permits you to use the bulk of the Visual Studio settings while selectively overriding as your needs require.

#### Annotations

The **Annotations** category of the Project Configuration dialog lets you control how annotations are stored and displayed. See page 184 for details on using annotations.



- **Author:** Type your name or the username you want to be associated with the annotations you create.
- Add Files: Click this button to browse for an existing annotation file (\*.ann). For example, you might want to add files created by other developers of this project so that you can see everyone's annotations. (If other developers are also annotating code using Understand, choose Annotations > Refresh Annotations from the menus when you want to get the latest annotations they have added.)
- Create File: Click this button to create a new annotation file. The default directory is the project directory.
- **Remove File:** Select a file and click this button to remove it from the list of files that are used to display annotations. Removing a file from this list does not delete the file from the file system.
- **Default file:** Select the file that should contain annotations you create.
- FG: Click on the colored block to change the text color for annotations in this file.
- **BG:** Click on the colored block to change the background color for annotations in this file.

- Show inline: When this box is checked, annotations are shown following the place where the associated entity is defined in the code. You can also turn this display feature on and off by choosing Annotations > Display Inline from the menus.
- Show in hover text: When this box is checked, annotations are shown if you point to a place where the associated entity is used in the code for 2 seconds. You can also turn this display feature on and off by choosing Annotations > Display hover text from the menus.
- Show indicator: When this box is checked, the entity has a squiggly line under it wherever it is used in the code. You can also turn this display feature on and off by choosing Annotations > Display indicator from the menus.

Annotations are stored in \*.ann files, which use the SQLite database format. In addition to viewing annotations in *Understand*, you can use other applications that support SQLite to modify and search annotation files.

Multiple projects can reference the same \*.ann files. Sharing annotation files across projects allows the same annotation to appear in multiple projects that share one or more source code files.

## **Ada Options**

In the Ada > Options category of the Project Configuration dialog, you can tell *Understand* how to analyze Ada source code. You see this window when you choose the **Project > Configure Project** menu item and select the Ada category.

Compiler —		
Version	Ada95 🗸	
Preprocessor	None	
Standard	conf\understand\ada\ada95 Reset	
Multiple Lan	nguage Linkage ————————————————————————————————————	
The case of	externally linkable entities is	
all lowers	case 💿 all uppercase	
Metrics		
Count and/	l/or operators in strict complexity	
Count exce	ception handlers in complexity	
Count for-I	loops in complexity	
Optimize —		
Create and	d cross reference record object components	
Create rela	ations between formal and actual parameters	
Less memo	nory usage versus speed	
Save com	ments associated with entities	
Options		
Display entity r	names as Original	
Prompt on	parse errors	
Main subprogra	rams: (main1, main2)	
Library Directo	ories Edit	

The fields in this category are as follows:

- Version: Choose the version of Ada used in your project. *Understand* supports Ada83, Ada95, Ada05, and Ada12.
- **Preprocessor:** Choose which type of preprocessor statements are used in your Ada code. The choices are None, C, Gnatprep, and Verdix. Note that if your source code directories contain a Gnat \*.gpr project file, that file will be analyzed whether or not you select the Gnatprep preprocessor.

• **Standard:** You may choose a directory that contains a standard library used by this project. Default standards are provided in <install\_directory>/conf/understand/ada.

Sometimes it is helpful to analyze code in context of its compilation environment rather than the environment defined as "Standard" in the Ada Language Reference Manual. This is most often needed when your compiler vendor offers bindings to other languages or low level attributes of a chip or system. To do so, place all the source files containing the Ada specifications for the new standard in one directory. Then point to this directory in the **Standard** field.

- **Case of externally linkable entities:** Choose which case should be used for "exporting" entities in this language that can be linked to (for example, called as functions) by other languages. For example, if an entity is declared in this language as "MYITEM" and you choose "all lowercase" here, other languages would be expected to call that entity as "myitem".
- Count and/or operators in strict complexity: Place a check in this box if you also want "and" and "or" operators considered when calculating the strict complexity metric shown in the Program Unit Complexity report. Strict complexity is like cyclomatic complexity, except that each short-circuit operator ("and then" and "or else") adds 1 to the complexity.
- **Count exception handlers in complexity:** If this box is checked (it is on by default), exception handlers are considered when calculating the complexity metrics shown in the Information Browser and the Program Unit Complexity report.
- **Count for-loops in complexity:** Remove the check from this box if you do not want FOR-loops considered when calculating the complexity metrics shown in the Information Browser and the Program Unit Complexity report. Complexity measures the number of independent paths through a program unit.
- Create and cross-reference record object components: If this box is checked (off by default), separate entities are created for components of all parameters and objects of a record type. By default, all references to object components are treated as references to the record type component.
- Create relations between formal and actual parameters: Place a check in this box if you want the analysis to create relations between formal and actual parameters. The actual parameters linked to formal parameters include items used in expressions passed as actual parameters. This option is off by default to speed up analysis.
- Less memory usage versus speed: Place a check in this box if you want to use Understand in a very low memory consumption mode. In order to conserve memory, Understand frees memory used to process a program unit if that program unit is not needed. Using this option may slow down operation significantly. It is off by default.
- Save comments associated with entities: Choose whether source code comments that occur before and after an entity should be associated with that entity.
- **Display entity names as:** Choose whether entity names should be displayed in *Understand* with the same case as the source code (original), all uppercase, all lowercase, only the first letter capitalized, or mixed case.

	• <b>Prompts on parse errors:</b> By default, you are prompted for how to handle errors that occur when analyzing files. When prompted, you may choose to ignore that error or all future errors. Turn this option off to disable this prompting feature. If you turned it off during analysis, but later want to turn error prompting back on, check it here.
	<ul> <li>Main subprograms: Provide a comma-separated list of the names of the main subprograms in the project.</li> </ul>
	• Library Directories: Type a directory path or click Edit to browse for the location of a directory that contains Ada libraries. Library files are analyzed as part of a project, but are not included in reports. All subdirectories of the directory you select will also be used to find libraries.
Ada > Macros Category	Ada code may contain conditional compiler instructions in pragma statements. For example:
	PRAGMA IF DEVICE == D129
	The supported pragmas are IF, IFDEF, ELSIF, ELSE, and ENDIF. These pragmas are similar to preprocessor directives such as #ifdef in C code.
	For <i>Understand</i> to successfully analyze your software it needs to know what macro definitions should be set. For more about ways to configure macro definitions, see

Using the Undefined Macros Tool on page 125 and the SciTools website. In the Ada > Macros category of the Project Configuration dialog, you can specify what macros to define for use with pragmas. You see this window when you choose the **Project > Configure Project** menu item and select the Ada category and the Macros

Dinderstand Project Co	onfiguration: C:\temp\M	MyUnderstandProject.u	db 💌
Languages	Macro	Definition	New
File Types File Options	DEBUG_LEVEL	70	Edit
Scheduled Activities	INSTRUMENT_CODE	G54	Remove
▷ Reports Visual Studio			Import
⊿ Ada			Export
Options Macros			
	]		
			OK Cancel

Understand 4.0 User Guide and Reference Manual

subcategory.

The Macros category lists macros and their optional definitions. Each macro may be edited or deleted. To define a macro, click **New**.

Define a	a new macro
Macro:	DEVICE
Definition:	D129
	OK Cancel

Type the name of the macro in the first field and the definition (if any) in the second field. Then click **OK**.

A macro must have a name, but the definition is optional. Macros that have no definition value are commonly used in conjunction with PRAGMA IFDEF statements to test whether a macro is defined.

To change the definition of an existing macro without changing the name, select the macro and click **Edit**.

You can import or export a list of macros and their optional definitions by clicking **Import** or **Export** and selecting the file. The file must contain one macro definition per line. A # sign in the first column of a line in the file indicates a comment. Separate the macro name and its definition with an equal sign (=). For example, *DEBUG=true*.

You can set macros on the und command line with the -define name[=value] option.

## **Assembly Options**

In the **Assembly > Options** category of the Project Configuration dialog, you can tell *Understand* how to analyze assembly source code. You see this window when you choose the **Project > Configure Project** menu item and select the **Assembly** category. Currently the only setting is the assembler, and Coldfire 68K is the only assembler supported.

The **Assembly > Includes** category in the Project Configuration dialog (which you open with **Project > Configure Project**) allows you to specify include directories for assembly code. You can specify multiple directories to search for include files used in the project.

Understand Project Configuration: C:\Users\Yvonne\AppData\Roaming\SciT				
Languages Files	Options			
File Types File Options	Assembly Includes	New		
Scheduled Activities	C:\software\test	Edit		
<ul> <li>Reports</li> </ul>		Remove		
Visual Studio		Move Up		
Options		Move Down		
VCTT		Import		
	System Include Path:			
	ок	Cancel		

Typically only include files that are not directly related to your project, and that you do not want to analyze fully are defined here. For project-level includes that you want to be analyzed, add those include files as source files in the **Files** category.

To add a directory, click the **New** button and then the ... button, browse to the directory, and click **OK**.

🔎 Add a A	ssembly Includes D	irectory	×
Directory	C.\aaftuuara\taat		
Directory:	Chsoftwareitest	•	
	ОК	Cancel	

During analysis, the include directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

For the **System Include Path**, browse to select the directory that contains system include files (include filenames surrounded by < >).

Include files found in regular include directories are added to the project. Include file found in system include directories are not added.

Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories.

You may use environment variables in include file paths. Use the \$var format on Unix and the %var% format on Windows. You can also use named root in include file paths (see page 107).

You can import a list of include directories from a text file by clicking **Import** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

#### **COBOL Options**

In the **COBOL > Options** category of the Project Configuration dialog, you can tell *Understand* how to analyze COBOL source code. You see this window when you choose the **Project > Configure Project** menu item and select the **COBOL > Options** category.

Files File Types File Options Scheduled Activities ▷ Metrics ▷ Reports Visual Studio ▲ COBOL Options Copybooks	Compiler Compiler AcuCobol Format Fixed
---	---

The field in the **COBOL > Options** category is as follows:

- **Compiler:** Select the compiler that you use. The options are Ansi85, MicroFocus, AcuCobol, IBM, HP OpenVMS, and Unisys.
- Format: Choose whether the source code is in fixed or free format.

#### COBOL > Copybooks Th Category op

The **COBOL** > **Copybooks** category in the Project Configuration dialog (which you open with **Project** > **Configure Project**) allows you to specify directories that contain files included with the COPY statement. Typically, such files have a \*.cpy file extension. You can specify multiple directories to search for such files used in the project.

Specify directories here if they contain files that are not directly related to your project, and that you do not want to analyze fully. For copybooks that you want to be analyzed, add those files as source files in the **Files** category.

Dunderstand Project Co	nfiguration: C:\Users\Yvonne\Ap	pData 💌
Languages	Copybook Paths	New
Files File Types File Options	C:\software\copybooks	Edit
Scheduled Activities		Remove
<ul> <li>▷ Metrics</li> <li>▷ Reports</li> <li>Visual Studio</li> </ul>		Move Up
COBOL Options		Move Down
Copybooks		Import
		Export
	Search for copybook files among project files	
	ок	Cancel

To add a directory, click the **New** button and then the ... button, browse to the directory, and click **OK**.

During analysis, the copybook directories are searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

If you check the **Search for copybook files among project files** box, your project directories will be searched along with any directories you specify here. When searching for a copybook, the search looks in the directories specified in this dialog first. It then searches among the project files if this box is checked.

Copybook paths are not recursively searched; that is, any subdirectories will not be searched for copybook files unless that subdirectory is explicitly specified in the list of copybook directories.

You may use environment variables in copybook file paths. Use the \$var format on Unix and the %var% format on Windows. You can also use named root in copybook file paths (see page 107).

You can import or export a list of copybook directories from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Relative paths are relative to the project file.)

### C++ (Fuzzy) Options

In the **C++** > **Options** category of the Project Configuration dialog, you can tell *Understand* how to analyze C and C++ source code. If you selected the **Fuzzy** option for C/C++ in the Languages category, you see the following window when you choose the **Project** > **Configure Project** menu item and select the **C++** category.

Compiler			
Compiler	Microsoft Visual C++ 💌		
Compiler include paths %include%			
Allow nested comments			
Multiple Language Linkage			
Prepend the names of ex	Prepend the names of externally linkable entities with		
Append the names of externally linkable entities with			
Optimize			
Create references in	inactive code		
Create references to	Create references to local objects		
Create references to	Create references to macros during macro expansion		
Create references to	Create references to parameters		
Create references in inline assembly			
Save comments associated with entities			
Save duplicate refer	Save duplicate references		
Save macro expansion	ion text		
Use include cache			

(If you selected the **Strict** option in the Languages category, see C++ (*Strict*) Options on page 75 for how to configure your project.)

The fields in the C++ > Options category are as follows:

- **Compiler:** Select the compiler/platform that you use. Many different compilers are supported. Your choice affects how *Understand* analyzes the project. Note that not all features of a particular compiler will necessarily be handled.
- **Compiler Include Paths:** Type the path the compiler uses to find include files. For example, %include%.
- Allow nested comments: By default, this is off. If turned on it permits C style (/\* \*/) comments to be nested. This isn't permitted by the ANSI standard, but some compilers do permit it.

- **Prepend the names of externally linkable entities with:** You may optionally type a string that you want used as a prefix to reference all linkable entities in other source code languages.
- Append the names of externally linkable entities with: You may optionally type a string that you want used as a suffix to reference all linkable entities in other source code languages.
- **Create implicit special member functions:** Check this box if you want a default constructor and destructor to be created in the database and given implicit declaration references, if they are not declared in the source code for class and struct entities. This option provides entities for the analyzer to reference when they are called. The default is off.
- Create references in inactive code: If you wish to exclude cross-reference information for code that is IFDEFed out by the current macro settings, turn this option off. By default, this option is on and cross-reference information for inactive code is included.
- Create references to local objects: By default, all local object declarations are included in the database. If you wish to exclude variables declared within functions from the database, turn this option off. Local objects included for analysis can then be either included or excluded from the HTML output generated. Specify whether to include local objects in the HTML output on the main window of *Understand*.
- Create references to macros during macro expansion: Checking this box causes references to be stored during macro expansion. In some cases, this is useful. Be aware that enabling this option can add many references and make the database large and slower. The default is off.
- **Create references to parameters:** If you wish to exclude cross-reference information for parameters, turn this option off. By default, this option is on and all cross-reference information for parameters is included.
- Create references in inline assembly: Check this box if you want crossreferences to be created to assembly code for any #asm preprocessor macros in your code.
- Save comments associated with entities: Choose whether source code comments that occur before and after an entity should be associated with that entity.
- **Save duplicate references:** By default, duplicate cross-references are condensed to a single cross-reference. To keep duplicates, check this box.
- Save macro expansion text: If you put a check in this box, you can right-click on a
  macro and choose Expanded Macro Text from the context menu to see how the
  macro expands.
- Use include cache: By default, include files are cached during the analysis phase as they are often referenced in multiple source files. This speeds up analysis, but also uses more memory. If you have problems with excessive memory use during analysis, turn this option off. Note that there are also situations where turning the include cache on or off can affect analysis results, particularly where include actions are dependent on where they are included.

# C++ > IncludesThe C++ > Include category in the Project Configuration dialog (which you open with<br/>Project > Configure Project) allows you to specify include directories. You can specify<br/>multiple directories to search for include files used in the project.

The configuration of your include file directories is important to improving the accuracy of project analysis. For more about ways to configure these directories, see *Using the Missing Header Files Tool* on page 123.

Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories.

To add a directory, click the **New** button and then the ... button, browse to the directory, and click **OK**.

🔎 Add a	C/C++ Includes Directory	×
Directory:	C:\MyPrograms\includes -	

During analysis, the include directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

Typically only include files that are not directly related to your project (such as systemlevel includes) and that you do not want to analyze fully are defined here. For projectlevel includes that you want to be analyzed, add those include files as source files in the **Files** category.

You may use environment variables in include file paths. Use the \$var format on Unix and the %var% format on Windows. You can also use named roots in include file paths (see page 107).

You can import or export a list of include directories from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

The **C++** > **Include** category provides the following options to control include handling:

- Add found include files to source list: Enabling this option causes include files found during project analysis to be added to the project automatically. This allows you to see more detailed information about such include files. The default is off.
- Add found system include files to source list: If you choose to add include files that are found to the source list, you can also choose whether system include files should be added. The default is off.
- **Prompt for missing include files:** If any include files cannot be found during analysis, you will normally see the **Include Paths** button in the Analysis Log after

	you analyze the project. If you want to be prompted for how to handle missing files during the analysis, you must choose <b>Tools &gt; Options</b> and enable the <b>Allow</b> <b>prompting for missing include files on a per project basis</b> checkbox in the <b>Analyze</b> category (page 105). Then, you will see this field in the Project Configuration dialog. If you then check the <b>Prompt for missing include files</b> box, you may choose to add a directory to the include path, ignore the missing file, or stop warning about missing files during the analysis.
	• Search for include files among project files: This option directs the analyzer to look among project files as a last resort for missing include files. The default is on.
	• <b>Treat system includes as user includes:</b> This option tells the analyzer to look for system includes (surrounded by < >) using the same strategies as normal includes (surrounded by quotes). If this item is off, the analyzer looks for system includes only in directories defined by the compiler configuration. The default is on.
	• <b>Ignore directories in include names:</b> Check this option if you want to ignore any directory specifications in #include statements and instead use the include file wherever it is found in the project. The default is off.
	• Use case-insensitive lookup for includes: This option tells the analyzer whether to ignore the case of filenames in #include statements. The default is off. (Not available on Windows; Windows lookups are always case-insensitive.)
C++ > Includes > Auto Category	In the <b>C++ &gt; Includes &gt; Auto</b> category you can specify include files that should be included before each file in a project.
	To add a file, click <b>New</b> and browse for the file(s). Then click <b>Open</b> .
	You can import or export a list of auto include files from a text file by clicking <b>Import</b> or <b>Export</b> and selecting the text file that contains one file path per line.
	Use the <b>Move Up</b> and <b>Move Down</b> buttons to change the order in which these files are included.
C++ > Includes > Ignore Category	In the <b>C++ &gt; Includes &gt; Ignore</b> category you can specify individual include files that you wish to ignore during analysis.
	To add a file to be ignored, click <b>New</b> and type the filename of the include file. Then click <b>OK</b> . The filename can use wildcards, such as moduleZ *.h, to match multiple files.

🦻 Ig	nore includes	? ×
Igno	ore include file:	
temp.h		
	ОК	Cancel

Any missing files you choose to ignore when prompted during analysis will be added to this list.

You can import or export a list of files to ignore from a text file by clicking **Import** or **Export** and selecting the text file that contains one filename per line.

#### .....

#### C++ > Includes > Replacement Text

In the **C++ > Includes > Replacement Text** category you can specify text that should be replaced in include file text.

For example, you might use this feature to replace VAX/VMS include paths like [sys\$somewhere] with valid Unix or Windows paths without modifying the source code.

To add an item, type the string found in the actual include files in the **Include String** field. Type the text you want to replace it with in the **Replace With** field. Then click **OK**.

🔎 Define a ne	w Replace include string
Include String	
Replace With	
	OK Cancel

You can import or export a list of include strings and their replacements from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one include string per line. The file should separate the include string and its replacement with an equal sign (=).

Use the **Move Up** and **Move Down** buttons to change the order in which these replacements are made.

#### .....

#### C++ > Macros

Category

C source code is often sprinkled with pre-processor directives providing instructions and options to the C compiler. Directives such as the following affect what the software does and how it should be analyzed:

#define INSTRUMENT\_CODE #ifdef INSTRUMENT\_CODE ... statements ... #endif

Macros are often defined with directives (#define) in include files (.h files) or are passed in via the compiler (typically with the -D option).

For *Understand* to successfully analyze your software it needs to know what macro definitions should be set. For more about ways to configure macro definitions, see *Using the Undefined Macros Tool* on page 125 and the SciTools website.

The C++ > Macros category in the Project Configuration dialog (which you open with **Project > Configure Project**) allows you to define preprocessor macros that are used when compiling the code.
To add a macro definition, click the **New** button and type the name of the macro and optionally a definition. Then click **OK**.

🔎 Define a	a new macro	<b>—</b> ×	
Macro:	DEBUG		]
Definition:			]
	ОК	Cancel	

Note that a macro must have a name, but that the definition is optional. Macros that are defined but have no definition value are commonly used in conjunction with *#ifdef* preprocessor statements to see if macros are defined.

**Note:** A number of preprocessor macros are automatically supported. In additions to the common macros, *Understand* supports the following macro formats for embedded assembly code if you are using the "fuzzy" analyzer. (The strict C/C++ analyzer does not support these macro formats.)

```
#asm(<embedded assembly code>);
#asm "<embedded assembly code>";
#asm
<embedded assembly code>
#endasm
```

You can import or export a list of macros and their optional definitions from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one macro definition per line. A # sign in the first column of a line in the file indicates a comment. The file should separate the macro name and its definition with an equal sign (=). For example, *DEBUG=true*.

The priority for macro definitions is as follows, from lowest to highest priority:

- 1 Built-in language macros (\_\_FILE\_\_, etc.)
- 2 Compiler configuration file
- 3 Macro definitions in a synchronized Visual Studio project
- 4 Undefines of compiler defines (via the **Configure Undefines** button)
- 5 Project defines (Macros category)
- 6 Define on und command line using -define
- 7 Define in source file (#define / #undefine in source)

- -----

C++ > Macros > Undefines Category You can list undefined macros in the **C++** > **Macros** > **Undefines** category in the Project Configuration dialog. Click **New** and type the name of a macro that you do not want to be defined. Then click **OK**.

🦻 Add a	an Undefine Macro	x
Macro:	F00	
	······	
	OK Cancel	

You can import or export a list of undefined macros from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one macro name per line. A # sign in the first column of a line in the file indicates a comment.

## C++ (Strict) Options

See *Languages Category* on page 41 for information about the differences between the default C/C++ analyzer and the strict analyzer.

**Note:** If you did not select the **Strict** option in the Languages category next to the C/C++ box, see C++ (*Fuzzy*) Options on page 68 for how to configure your project.

In the C++ (Strict) > Options category of the Project Configuration dialog, you can control how C/C++ source code is analyzed. You see this window when you choose the **Project** > Configure Project menu item and select the C++ (Strict) category.

Target			
Arch	x86_64 ▼		
Vendor	pc 🔹		
OS	win32 🔹		
Env	unknown		
Version	1300		
	Delayed Template Parsing		
Language Standard			
C C8	9 🔹		
C++ C+	+98 🔹		
Optimize			
Create references in inactive code			
Save comments associated with entities			
Save macro expansion text			
Warnings: 🖲 None 🔘 Default 🔘 All			
Objectiv	e-C		
Memory Ma	anagement  MRR  ARC  GC		

The first three fields in the Target section of this dialog match target triplets used by the GNU Compiler Collection (GCC). The defaults match the platform on which you are running *Understand*. These fields are used to control which extensions (such as preprocessor defines, header search paths and language syntax) are analyzed. If your choices here do not match the code used, errors are likely to occur during the analysis.

If your code is built for multiple targets, use these options to switch between target environments for the code analysis.

The fields in the C++ (Strict) > Options category are as follows:

- Arch: Select the architecture of the chip for which your project is written. Examples of the many supported options include ARM, PowerPC64, and x86\_64.
- Vendor: Select the source of the chip architecture. Examples include Unknown, Apple, PC, and SCEI (Sony PlayStation). Use the "Unknown" option to select the most generic C/C++ code analysis.
- **OS:** Select the operating system that this program will be used under. Examples include iOS, Linux, and Win32.
- Env: Select the build environment you use to build this project. Examples include GNU, EABI, and Mach-O. For most projects, the default of "unknown" is fine.
- Version: Specify the appropriate version number. If your OS is IOS or MacOSX, specify the operating system version number. If your OS is Win32, specify the Microsoft C (MSC) version of your compiler. For example, specify 1300 for Visual C++ .NET and 1700 for Visual C++ 2012.
- **Delayed Template Parsing:** If your OS is Win32, you can choose whether to delay parsing of template files. This option is required for compatibility with MSVC. However, be aware that unreferenced template code will not be analyzed at all if you enable delayed template parsing.
- C Language Standard: Select the C standard to which you want your C code to conform.
- C++ Language Standard: Select the C++ standard to which you want your C++ code to conform.
- Create references in inactive code: If you wish to exclude cross-reference information for code that is IFDEFed out by the current macro settings, turn this option off. By default, this option is on and cross-reference information for inactive code is included.
- Save comments associated with entities: Choose whether source code comments that occur before and after an entity should be associated with that entity.
- Save macro expansion text: If you put a check in this box, you can right-click on a
  macro and choose Expanded Macro Text from the context menu to see how the
  macro expands.
- Warnings: Choose how many of the warnings provided by the strict analyzer you want reported. These warnings indicate potential problems in the source code. Choosing to see some or all warnings is likely to slow down the project analysis somewhat.
- **Memory Management:** If you use Objective-C, select the memory management mode used. The options are MMR (manual retain-release), ARC (automatic reference counting), and GC (garbage collection).

#### C++ (Strict) > Includes Category

The **C++ (Strict) > Includes** category in the Project Configuration dialog (which you open with **Project > Configure Project**) allows you to specify include directories. You can specify multiple directories to search for include files used in the project.

The configuration of your include file directories is important to improving the accuracy of project analysis. For more about ways to configure these directories, see *Using the Missing Header Files Tool* on page 123 and the SciTools Support website.

Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories.

To add a directory, click the **New** button and then the ... button, browse to the directory, and click **OK**.

🔎 Add a Incl	udes Directory	×
Directory:		
	OK Cancel	

During analysis, the include directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

Typically only include files that are not directly related to your project (such as systemlevel includes) and that you do not want to analyze fully are defined here. For projectlevel includes that you want to be analyzed, add those include files as source files in the **Files** category.

You may use environment variables in include file paths. Use the \$var format on Unix and the %var% format on Windows. You can also use named roots in include file paths (see page 107).

You can import or export a list of include directories from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

The **C++ (Strict) > Include** category provides the following options to control how includes are handled:

• **Sysroot:** This field allows you to specify the root of the default header search path. For example, if you set Sysroot to /dir, the analyzer searches /dir/usr/include instead of /usr/include. This is useful if you use cross compilers or builds against a different SDK from the host machine. This option corresponds to the --sysroot command line option in compilers such as gcc, icc, and clang. This option is available for all supported platforms.

	<ul> <li>Add found include files to source list: Enabling this option causes include files found during project analysis to be added to the project automatically. This allows you to see more detailed information about such include files. The default is off.</li> </ul>
	<ul> <li>Add found system include files to source list: If you choose to add include files that are found to the source list, you can also choose whether system include files should be added. The default is off.</li> </ul>
	<ul> <li>Prompt for missing include files: If any include files cannot be found during analysis, you will normally see the Include Paths button in the Analysis Log after you analyze the project. If you want to be prompted for how to handle missing files during the analysis, you must choose Tools &gt; Options and enable the Allow prompting for missing include files on a per project basis checkbox in the Analyze category (page 105). Then, you will see this field in the Project Configuration dialog. If you then check the Prompt for missing include files box, you may choose to add a directory to the include path, ignore the missing file, or stop warning about missing files during the analysis.</li> </ul>
	<ul> <li>Search for include files among project files: This option directs the analyzer to look among project files as a last resort for missing include files. The default is on.</li> </ul>
	<ul> <li>Ignore directories in include names: Check this option if you want to ignore any directory specifications in #include statements and instead use the include file wherever it is found in the project. The default is off.</li> </ul>
	• <b>Compare files by content instead of path:</b> Check this option if you want include files to be compared by their contents rather than by their file path. The default is off.
	There are a number of additional options for include file handling that are available only if you are using the default analyzer rather than the strict analyzer.
C++ (Strict) > Includes >	In the <b>C++ (Strict) &gt; Includes &gt; Frameworks</b> category lets you specify Mac OS and iOS framework paths that the project uses.
Frameworks	To search a directory and its subdirectories for a framework, click Search.
Category	To add a location, click <b>New</b> and browse for the folder. Then click <b>Select Folder</b> and then <b>OK</b> . You can import or export a list of framework folders from a text file by clicking <b>Import</b> or <b>Export</b> and selecting the text file that contains one path per line.
	Use the <b>Move Up</b> and <b>Move Down</b> buttons to change the order in which these folders

are processed.

C++ (Strict) > Includes > Prefix Headers Category	A prefix header is a C/C++ header file that is included at the beginning of every source file by the compiler. This is done without the use of a #include directive. It is common for Mac OS X programs to use prefix header files.
	In the <b>C++ (Strict) &gt; Includes &gt; Prefix Headers</b> category, you can specify files that are used as prefix header files.
	To add a file, click <b>New</b> and browse for the file. Then click <b>Open</b> . You can import or export a list of files from a text file by clicking <b>Import</b> or <b>Export</b> and selecting a text file that contains one file path per line.
	Use the <b>Move Up</b> and <b>Move Down</b> buttons to change the order in which these files are processed.
C++ (Strict) > Macros Category	For information about the <b>C++ (Strict) &gt; Macros</b> category, see C++ > Macros Category on page 72.
	For information about the <b>C++ (Strict) &gt; Macros &gt; Undefines</b> category, see C++ > <i>Macros &gt; Undefines Category</i> on page 74.

## **C# Options**

In the **C# > Options** category of the Project Configuration dialog, you can control how C# source code is analyzed. You see this window when you choose the **Project > Configure Project** menu item and select the **C#** category.

Save comments associated with entities

Analyze found reference files

The fields in the C++ (Strict) > Options category are as follows:

- Save comments associated with entries: Choose whether source code comments that occur before and after an entity should be associated with that entity. The default is on.
- Analyze found reference files: If this box is unchecked, methods in reference libraries are not counted for the purpose of computing metrics. The default is on.

In the **C# > References** category, click **New**. Click ... and browse for a \*.dll file. Type the alias for that file used in the code and click **OK**.

🧏 Def	fine a new Reference
File	C:\MyFiles\myutil.dll
Alias	myutil
	OK Cancel

You can import or export a list of reference files and their aliases from a text file by clicking **Import** or **Export** and selecting a file that contains one reference and its alias per line. The file should separate the reference file and its alias with an equal sign (=).

By default, reference files are analyzed as part of the project. If you do not want them to be analyzed, uncheck the **Analyze found reference files** box in this category. If this box is unchecked, methods in reference libraries are not counted for the purpose of computing metrics.

In the **C# > Preprocessor Symbols** category, you can click **New** to add symbol names that should be treated as defined when analyzing preprocessor directives such as #if.

## **Fortran Options**

In the **Fortran > Options** category of the Project Configuration dialog, you can specify how to analyze Fortran source code. You see this window when you choose the **Project > Configure Project** menu item and select the **Fortran** category.

Compiler
Version Fortran2003 -
Format Options
Format Auto
Truncate column 72 Column: 72
Free format file filters
Allow C-style comments
Allow ;* comments
Allow colons in names
Allow function declaration without parentheses
Allow parameter declaration without parentheses
Allow quote in octal constants
Case sensitive identifiers
Use preprocessor
Intrinsics File C:\Program Files\SciTools\conf\understand\fortran\intrinsics03.txt Reset
Multiple Language Linkage
The case of externally linkable entities is
Ill lowercase
Prepend the names of externally linkable entities with
Append the names of externally linkable entities with
Options
V Prompt on parse errors
Display entity names as Original 👻

The fields in the **Fortran > Options** category are as follows:

- Version: Select the variant of Fortran used by the source code in this project. If you change the version after creating a project, the project will be reanalyzed when you click OK. The choices are Fortran77, Fortran90, Fortran95, and Fortran2003. If you have a mix of code, choose the newest language variant. That is, if you have F77 and F95 code, choose F95. The default is Fortran95.
- Format: Some older Fortran variants and all new variants permit *free form* statements, which may cross lines). Fixed form statements are terminated by a line end or column number. The default is "auto format," which automatically detects the

parsing format (fixed or free) on a file-by-file basis. This allows you to mix free and fixed format. Auto format also determines the correct truncation point for fixed format files. Choose "fixed" or "free" only if all your source files have the same format. Blocks of freeform code can be used within a fixed format file if you bracket the blocks with !dec\$freeform and !dec\$nofreeform.

- **Truncate column:** If you choose fixed form, you may choose what column terminates statements. Common columns 72 and 132 are available or you may specify a column or no truncation.
- Allow C-style comments: Check this option if your Fortran code contains comments of the form /\* ... \*/.
- Allow ;\* comments: Allow the use of end-of-line comments that begin with ;\*
- Allow colons in names: Check this box to allow colons (:) to be used in identifiers in F77 code. Enabling this option could cause problems in F77 code that does not use this extension, so the default is off.
- Allow function declaration without parentheses: Check this box if you want to allow functions to be declared without the use of parentheses. By default, parentheses are required.
- Allow parameter declaration without parentheses: Check this box if you want to allow parameters to be declared without the use of parentheses. By default, parentheses are required.
- Allow quote in octal constants: Check this box if a double quote mark (") should be treated as the start of a DEC-style octal constant. For example, "100000. If this box is not checked (the default), a double quote mark begins a string literal.
- **Case sensitive identifiers:** Check this box if you want identifier names to be treated case-sensitively. By default, case is ignored.
- Use preprocessor: Use this option to disable or enable preprocessor support.
- Intrinsics file: Type or browse for a file that contains intrinsic functions you want to be analyzed. Default intrinsics files are provided in the <install\_directory>/conf/understand/fortran directory: intrinsics77.txt, intrinsics90.txt, and intrinsics95.txt.
- **Case of externally linkable entities:** Choose which case should be used for "exporting" entities in this language that can be linked to (for example, called as functions) by other languages. For example, if an entity is declared in this language as "MYITEM" and you choose "all lowercase" here, other languages would be expected to call that entity as "myitem".
- **Prepend the names of externally linkable entities with:** You may optionally type a string that you want used as a prefix to reference all linkable entities in other source code languages.
- Append the names of externally linkable entities with: You may optionally type a string that you want used as a suffix to reference all linkable entities in other source code languages.

	<ul> <li>Prompt on parse errors: By default, a prompt asks how to handle any parsing errors. When prompted during analysis, you may choose to ignore that error or all future errors. Turn this option off to disable this prompting feature. If you turned it off during analysis, but later want to turn error prompting back on, check it here.</li> <li>Display entity names as: Choose whether entity names should be displayed in <i>Understand</i> with the same case as the source code (original), all uppercase, all lowercase, only the first letter capitalized, or mixed case.</li> </ul>
Fortran>Includes Category	The <b>Fortran &gt; Includes</b> category in the Project Configuration dialog (which you open with <b>Project &gt; Configure Project</b> ) allows you to specify include directories. You can specify multiple directories to search for include files used in the project.
	The configuration of your include file directories is important to improving the accuracy of project analysis. For more about ways to configure these directories, see <i>Using the Missing Header Files Tool</i> on page 123 and the SciTools Support website.
	Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories.
	To add a directory, click the <b>New</b> button and then the button, browse to the directory, and click <b>OK</b> .
	During analysis, the include directories will be searched in the order that they appear in the dialog. You can click <b>Move Up</b> or <b>Move Down</b> to change the order in which directories will be searched.
	Typically only include files that are not directly related to your project (such as system- level includes) and that you do not want to analyze fully are defined here. For project- level includes you want analyzed, add those include files as source files in the <b>Files</b> category.
	You can import or export a list of include directories from a text file by clicking <b>Import</b> or <b>Export</b> and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)
	For more information, see C++ > Includes Category on page 70.
Other Fortran Categories	For information about the <b>Fortran &gt; Includes &gt; Replacement Text</b> category, see C++ > <i>Includes &gt; Replacement Text</i> on page 72.
	For information about the <b>Fortran &gt; Macros</b> category, see <i>C</i> ++ > <i>Macros Category</i> on page 72. The following predefined macros are supported:LINE,FILE,DATE, andTIME

## **Java Options**

In the **Java > Options** category of the Project Configuration dialog, you can specify how to analyze Java source code. You see this window when you choose the **Project > Configure Project** menu item and select the **Java** category.

Compiler
Version Java5 -
Metrics
Count Javadoc comments in line count metrics
Multiple Language Linkage
Prepend the names of JNI/KNI external entities with Java_
Include package name
Optimize
Save comments associated with entities

- Version: Select the version of Java used by the source code in this project. If you change the version after creating a project, the project will be reanalyzed when you click OK. The choices are Java 1.3, 1.4, 5, 6, 7, and 8.
- Count Javadoc comments in line count metrics: If this box is checked, Javadoc comments are included when computing the CountLine, CountLineComment, and RatioCommentToCode metrics. The default is on.
- **Prepend the names of JNI/KNI external entities with:** You can specify a prefix used by Java to call functions in other languages. A Java call to a function "func" would match the C function *prepend\_pkg\_class\_*func, where *prepend* is the string you specify here, *pkg* is the Java package name, and *class* is the Java class. This follows the Java Native Interface (JNI) and the Kaffe Native Interface (KNI).
- **Include package name:** By default, the package name is included in the prefix used to call functions in other languages. Uncheck this box to remove the package name from the names of external functions.
- Save comments associated with entities: Choose whether source code comments that occur before and after an entity should be associated with that entity.

## Java > Class PathsThe Java > Class Paths category allows you to identify Java .jar and .class files thatCategoryprovide classes for which you do not have source code.

Both .jar files and .class files are supported. Jar files contain compressed .java (source) files. Class files contain compiled sources. By default, the src.jar (or src.zip) file provided by the Java Developers Kit is located. You can add other .jar files as needed.

To add a directory with .class and .java files, follow these steps:

1 Click New Path.

- 2 Locate and select the directory containing .class files. You can provide a relative path to a directory by typing the path directly in the Class Path field rather than browsing for a directory.
- 3 Click OK.

🔎 Add a Cl	ass path	×
Class path	Cim/Code -	
Class path	C.mycode +	
	OK Cancel	

To add a .jar file to the list, follow these steps:

- 1 Click New Jar.
- 2 Locate and select the .jar or .zip file. You can select multiple .jar files while holding down the Ctrl key. You can provide a relative path to a file by typing the path directly in the Jar File field rather than browsing for a file.
- 3 Click Open.

If a class is found in both a .java and .class file in the class path, the class in the .java file is used.

You can import or export a list of class paths and/or jar files from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory or file path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

#### .....

If you use the Eclipse IDE for code development, you can access a number of *Understand* features within the Eclipse IDE by installing the Understand plugin for Eclipse. These features include the Entity Filter, Information Browser, Metrics, Treemaps, Butterfly graphs and Control Flow graphs. See http://scitools.com/eclipse/ for details on installing this plugin.



**Eclipse Plugin** 

## **JOVIAL Options**

In the **Jovial > Options** category of the Project Configuration dialog, you can specify how to analyze JOVIAL source code. You see this window when you choose the **Project > Configure Project** menu item and select the **Jovial** category.

Compiler		
Version	Jovial73 🔻	
Truncate column	None Column: 72	
Automatic compoo	I file	
Implementation		
Bits In Byte	8	
Bits In Pointer	16	
Bits In Word	16	
Fixed Precision	15	
Float Exp Bits	8	
Float Precision	23	
Int Precision	15	
Multiple Language Linkage ————————————————————————————————————		
The case of exte	rnally linkable entities is	
Il lowercase I all uppercase preserved		
Options		
Display entity names as Original		

The **Jovial > Options** category contains the following fields:

- Version: Select the JOVIAL version you use. JOVIAL73 and JOVIAL3 are supported.
- **Truncate column:** By default, statements are not truncated by column location. You may choose to truncate statements at column 72 or at some other user-defined column.
- Automatic compool file: Click ... and browse to the compool file you want to use. The file extension can be \*.txt, \*.cpl, or \*.jov. The selected file is automatically imported into all other files in the project.
- Implementation fields: The fields in this section allow you to specify the sizes and precision of various datatypes. These sizes vary with different implementations of JOVIAL. The sizes are used to determine data overlay. You can specify the number

of bits in a byte, number of bits in a pointer, number of bits in a word, precision for fixed datatypes, number of bits in a floating exponent, precision for floating datatypes, and the precision for an integer.

- **Case of externally linkable entities:** Choose which case should be used for "exporting" entities in this language that can be linked to (for example, called as functions) by other languages. For example, if an entity is declared in this language as "MYITEM" and you choose "all lowercase" here, other languages would be expected to call that entity as "myitem".
- **Display entity names as:** Choose whether entity names should be displayed in *Understand* with the same case as the source code (original), all uppercase, all lowercase, only the first letter capitalized, or mixed case.

Jovial > !Copy Category

The **Jovial** > **!Copy** category in the Project Configuration dialog (which you open with **Project** > **Configure Project**) lets you select directories to be searched for files named in !COPY directives.

To add a directory to the list, follow these steps:

1 Click the New.

🧏 Add a !(	Copy Directory		×
Directory:	C:\mycode\testbed	•	
	ОК	Cancel	

- 2 Click the ... button and browse to the directory you want to add.
- 3 Click OK.

When a !COPY directive is analyzed, the directories are searched in the order listed. To change the search order, select a directory and click **Move Up** or **Move Down**.

You can import or export a list of directories to be searched for files named in !COPY directives from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

## **Pascal Options**

In the **Pascal > Options** category of the Project Configuration dialog, you can specify how to analyze Pascal source code. You see this window when you choose the **Project > Configure Project** menu item and select the **Pascal** category.

Compiler		
Version Delphi 🔹		
Allow embedded SQ	L	
Predeclared entities file	$\label{eq:c:Program Files} C: \label{eq:conf} C: $	
dfm converter exe		
Multiple Language I	Linkage	
The case of externally Preserve	linkable entities is	
Options		
Display entity names as	Original 💌	

The **Pascal > Options** category contains the following fields:

- Version: Select the version of Pascal used by the source code in this project. The choices are Delphi, Compaq/DEC/HP, and Turbo. Select Compaq for legacy DEC Pascal projects. *Understand* supports all versions of Embarcadero's Delphi language and Embarcadero's Turbo Pascal language. It also supports ISO 7185:1990 (also known as Unextended Pascal) with HP Pascal extensions.
- Allow embedded SQL: Check this box to enable parsing of embedded SQL statements in your source code. Ingres embedded SQL statements are supported.
- Predeclared entities file: Click ... to select a text file (\*.txt) that contains predeclared routines, types, constants, and parameters used in your source code. Two versions of this file are provided in the <install\_directory>/conf/understand/pascal directory: predeclared.txt and predeclareddelphi.txt. The default is set according to your choice in the Version field.
- dfm converter exe: Browse for and select the executable to be used to convert binary Delphi Form (DFM) files in the project to text files. The text files will then be analyzed as part of the project. A number of third-party converters are available; Understand does not provide a converter.

	• <b>Case of externally linkable entities:</b> Choose which case should be used for "exporting" entities in this language that can be linked to (for example, called as functions) by other languages. For example, if an entity is declared in this language as "MYITEM" and you choose "Lowercase" here, other languages would be expected to call that entity as "myitem".
	• <b>Display entity names as:</b> Choose whether entity names should be displayed in <i>Understand</i> with the same case as the source code (original), all uppercase, all lowercase, only the first letter capitalized, or mixed case.
Pascal > Macros Category	The <b>Pascal &gt; Macros</b> category allows you to add support for preprocessor macros in source code. For example, the \$IF, \$IFDEF, and \$ELSE directives are supported.
	The CPU386 and MSWINDOWS macros are predefined for some types of Pascal/Delphi sources to avoid generating syntax errors with the standard library.
	For more information about the <b>Pascal &gt; Macros</b> category, see <i>C</i> ++ > <i>Macros Category</i> on page 72.
Pascal > Namespaces Category	The <b>Pascal &gt; Namespaces</b> category allows you to add a directory of namespaces to use when locating a unit specified in a USES statement. A USES statement may refer to a unit without specifying a namespace. So, directories you add in this category are searched in the order provided to find units with unspecified namespaces.
	For example, in the following statement, Unit1 has a namespace specified, so only the namespace CompanyName.ProjectName is searched for Unit1. Since Unit2 has no namespace specified, the namespaces in the Namespaces category will be searched for Unit2.
	uses CompanyName.ProjectName.Unit1, Unit2;
	To add a namespace directory, follow these steps:
	1 Click the <b>New</b> button.
	2 Click the button and browse to a directory. Then click <b>OK</b> .
	3 You can click <b>Move Up</b> or <b>Move Down</b> to change the precedence order in which the standard libraries are checked.
	You can import or export a list of directories to use when locating units from a text file by clicking <b>Import</b> or <b>Export</b> and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)
Pascal > Standard Library Paths	The <b>Pascal &gt; Standard Library Paths</b> category allows you to specify directories that should be searched for standard libraries.
Category	Standard library paths are used to find units that are not found in the project files. Only files that contain the required units are processed. For example, the following statement causes the standard libraries to be searched for a unit names System:
	Uses System; The standard libraries are not used when computing project metrics.

To add a directory, follow these steps:

- 1 Click the **New** button.
- 2 Click the ... button and browse to a directory. Then click **OK**.
- 3 You can click **Move Up** or **Move Down** to change the precedence order in which the standard libraries are checked.

You can import or export a list of directories that should be searched for standard libraries from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line.

.....

Pascal > Search PathsThe Pascal > Search Paths category allows you to specify directories to search for<br/>include files. To add a directory, follow these steps:

- 1 Click the New button.
- 2 Click the ... button and browse to a directory. Then click **OK**.
- **3** You can click **Move Up** or **Move Down** to change the precedence order in which the standard libraries are checked.

You can type a list of directory paths separated by semicolons.

You can import or export a list of directories to search from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line.

## **PL/M Options**

In the **PL/M > Options** category of the Project Configuration dialog, you can specify how to analyze PL/M source code. You see this window when you choose the **Project > Configure Project** menu item and select the **PL/M** category.

Compiler
Version PL/M-80 -
Options
Display entity names as Original ▼

The **PL/M > Options** category contains the following fields:

- Compiler Version: Choose the version of PL/M your compiler uses. The choices are PL/M-80 and PL/M-86.
- **Display entity names as:** Choose whether entity names should be displayed in *Understand* with the same case as the source code (original), all uppercase, all lowercase, only the first letter capitalized, or mixed case.

#### PL/M>Includes The **PL/M > Includes** category in the Project Configuration dialog (which you open with Project > Configure Project) allows you to specify include directories. You can specify Category multiple directories to search for include files used in the project. The configuration of your include file directories is important to improving the accuracy of project analysis. For more about ways to configure these directories, see Using the Missing Header Files Tool on page 123 and the SciTools Support website. Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories. To add a directory, click the **New** button and then the ... button, browse to the directory, and click OK. During analysis, the include directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched. Typically only include files that are not directly related to your project (such as systemlevel includes) and that you do not want to analyze fully are defined here. For projectlevel includes you want analyzed, add those include files as source files in the Files category. You can import or export a list of include directories from a text file by clicking Import or Export and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.) For more information, see C++ > *Includes Category* on page 70.

For information about the **PL/M > Includes > Replacement Text** category, see C++ >

Includes > Replacement Text on page 72.

## **Python Options**

In the Python > Options category of the Project Configuration dialog, you can specify how to analyze Python source code. You see this window when you choose the Project > Configure Project menu item and select the Python > Options category.

Version	
Python executable	C:\Program Files (x86)\Python\python.exe
Version: Pytho	n3
Python2	
Python3	
Standard Library	
🔽 Use built-in standard	l library files

The Version section lets you specify the version of Python to use. You can select one of the following:

	• <b>Python executable.</b> Click and browse for the location of the file you use to run Python programs. The version of this executable will be detected and displayed. In addition, the Python interpreter's sys.path variable is examined to find the directories to be searched for modules. The <b>Default</b> button fill in a Python executable path if one is found in the PATH environment variable definition.
	• <b>Python 2.</b> Select this option to analyze the project using the Python 2 standard.
	• <b>Python 3.</b> Select this option to analyze the project using the Python 3 standard.
	<b>Use Built-in Standard Library Files:</b> By default, <i>Understand</i> uses the files in the ./conf/understand/python/python2 or ./conf/understand/python/python3 subdirectory of the <i>Understand</i> installation to resolve Python standard library entities. These files contain stubs for such entities. You can disable this part of the search here.
	If a module is found in both the built-in libraries ( <i>Understand</i> 's copy) and the installed Python libraries, the version in the installed libraries takes precedence.
Python > Imports Category	The <b>Python &gt; Imports</b> category in the Project Configuration dialog (which you open with <b>Project &gt; Configure Project</b> ) allows you to specify import directories. You can specify multiple directories to search for import files used in the project.
	Import paths are not recursively searched; that is, any subdirectories will not be searched for import files unless that subdirectory is explicitly specified in the list of import directories.
	To add a directory, click the <b>New</b> button and then the button, browse to the directory, and click <b>OK</b> .

During analysis, the import directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

Typically only import files that are not directly related to your project and that you do not want to analyze fully are defined here. For project-level imports you want analyzed, add those files as source files in the **Files** category.

You can import or export a list of directories from a text file by clicking **Import** or **Export** and selecting the file. The file must contain one directory path per line. (In all such imported text files, a # sign in the first column of a line in the file indicates a comment. Full or relative paths may be used. Any relative paths are relative to the project file.)

For more information, see C++ > *Includes Category* on page 70.

## **VHDL Options**

There is currently no Project Configuration category for VHDL.

If you are new to *Understand*, you should be aware that the following terms have different meanings in *Understand* than they do in VHDL:

- Entity. Any source construct such as a file, function, or variable. This also includes, but is not limited to, VHDL entities.
- Architecture. An arbitrary collection of Understand entities organized in a hierarchy. This collection may contain, but is not limited to, VHDL architectures.

### **Web Options**

In the **Web** category of the Project Configuration dialog, you can specify what types of tags to allow in PHP files that are part of the project. You see this window when you choose the **Project > Configure Project** menu item and select the **Web** category.

Javascript Options	
Analyze Node.js	
Search path	
Predefined script\nodejs_predefined.txt Reset	
Search Strings for Entity Names	
PHP Options	
Version 5.3 -	
Allow Short Tags	
Allow ASP Style Tags	
Save comments associated with entities	

Web languages included in the analysis include CSS, HTML, JavaScript, PHP, and XML. For some file types, such as XML, only line count metrics are generated.

The Web category contains the following fields:

- Analyze jQuery: Check this box if you want the analysis to interpret a \$(...) call as a jQuery call. A JQuery Selector entity is created for the string literal parameter passed to "\$". For example, \$(".foo") would create a JQuery Selector entity named ".foo". Also, a jQuery Selector Uses category is added to the Information Browser for files and functions that use the jQuery selectors. By default, this option is off.
- Analyze Node.js: Check this box if you want the analysis to recognize the node.js "require" function. A **Requires** category is added to the Information Browser for files that contain calls to "require" and a **Required by** category is added to the Information Browser for files that are named in a "require" call. Requires and Required By graph views are also available.

The **Search Path** field allows you to specify a semicolon-separated list of directories to be searched in the order given for files named in "require" calls.

The **Predefined** field allows you to specify the name of a file containing a list of predefined node.js modules. This field defaults to the path to scitools\conf\understand\javscript\nodejs\_predefined.txt.

- Search Strings for Entity Names: Check this box if you want strings within JavaScript code to be searched for references to entities.
- **PHP Version:** Select the version of PHP used by your project. The options are 5.3, 5.4, 5.5, and 5.6.
- Allow Short Tags: Check this box if your PHP code ever uses the short form of PHP tags.
- Allow ASP Style Tags: Check this box if your PHP: Hypertext Preprocessor (PHP) code ever uses Active Server Pages (ASP) style tags.
- Save comments associated with entities: Choose whether source code comments that occur before and after an entity should be associated with that entity.

## **Setting General Preferences**

Understand allows you to control a number of aspects of its operation using the Understand Options dialog. To open this dialog, choose **Tools > Options**. This dialog provides options to set in the categories shown to the right:

The subsections that follow describe each of the categories:

- General Category on page 96
- User Interface Category on page 98
- User Interface > Lists Category on page 99
- User Interface > Alerts Category on page 100
- User Interface > Windows Category on page 101
- User Interface > Application Styles Category on page 103
- Key Bindings Category on page 104
- Analyze Category on page 105
- Configure Category on page 106
- Command Window Category on page 106
- Portability Category on page 107
- Dependency Category on page 108
- Editor Category on page 109
- Editor > Advanced Category on page 111
- Editor > Macros Category on page 114
- Editor > Styles Category on page 115
- Editor > Navigation Category on page 116
- Editor > External Editor Category on page 117
- Graphs Category on page 118



General Category	The following options can be controlled from the <b>General</b> category of the <b>Tools &gt;</b> <b>Options</b> dialog:
	Application Font
	Font : Arial Size : 8 Change Font
	Show on Startup
	$\blacksquare$ Show the Splash-Screen on startup $\blacksquare$ Show the Getting Started Dialog on startup
	Auto Loading/Saving Options
	Save all modified editor windows when application loses focus.
	Open last project on startup (currently: C:\sample\fastgrep\fastgrep.ud
	Use Default Working Directory
	C:/Users/Yvonne/AppData/Roaming/SciTools/sample/zlib Browse
	Performance
	Enable permissions checking for NTFS filesystems     Auto-detect network proxy settings
	Allow interactivity during intensive processing
	Allow events processing every 100 👘 milliseconds.
	Application Data
	Location: C:/Users/Yvonne/AppData/Roaming Browse
	Restart the application for this setting to take effect.

- Application font: To change the font used in dialogs and lists in *Understand*, click Change Font and select the font, font style, and font size you want to use and click OK.
- Show the Splash-Screen on startup: If checked (on by default), the logo is shown while *Understand* is starting.
- Show the Getting Started dialog on startup: If checked (on by default), the Getting Started tab (see page 21) is shown in the document area when you start *Understand*.
- Save all modified editor windows when application loses focus: If checked (off by default), then whenever you move to another application, any editor windows in which you have made changes have their contents saved automatically.
- **Open last project on startup:** If checked (off by default), the most recently opened project is automatically opened when you start *Understand* with no other project specified. This is a useful option if you typically work with only one project.

- Use default working directory: If checked (off by default), you can select an alternate default directory. This will be the starting place when you are browsing for other directories and the directory to which relative directory specifications relate. The default is the directory where your project is saved.
- Enable permissions checking for NTFS filesystems: If you check this box, file permissions are checked on NTFS filesystems when you use the editor to modify files. This option is off by default, since this checking can significantly degrade performance in some cases.
- Auto-detect network proxy settings: If the Getting Started tab does not show the Scientific Toolworks blog feed, you can check this option to have your system's proxy settings checked so that the feed can be loaded. However, scanning for proxy settings takes some time and slows down the *Understand* startup process. This option is disabled by default. After you enable this option, you will need to restart *Understand* in order for it to take effect.
- Allow interactivity during intensive processing: If checked (on by default), you can interact with *Understand* while it is performing background processing. Your interactive events are processed at the interval you specify in milliseconds.
- Allow events processing every n milliseconds: Specify how often interactive events are processed. By default, such events are processed every 100 milliseconds (0.1 seconds). You can improve background processing performance by reducing this value.
- Application Data Location: This field shows where files used internally by *Understand* but not associated with a specific project are stored. You can browse to change this location. You will need to restart *Understand* to have changes to this directory location take effect.

User Interface The following options can be set from the **User Interface** category of the **Tools** > Options dialog: Category Animate Windows/Drawers Speed Document Area Show tabs Use "Most Recently Used" order for next tab activation when documents close Dock Window Layouts Dock windows auto-hide to give maximum space for Tight Layout working in the Document Area Classic Layout Docks are laid out as in Understand 1.4 Multi-monitor Layout Docks are laid out on a separate screen Title Formats Window Titles: Long Ŧ Tab Titles: Short Ŧ Selector Files: Short Ŧ

- Animate Windows/Drawers: If checked (on by default), opening and closing windows and tabbed areas (drawers) is animated. You can choose a faster or slower speed than the default.
- **Show tabs:** If checked (the default), tabs are shown at the top of the document area for each of the windows open in that area. This includes the source editor windows, graphical views, and other windows.
- Use "Most Recently Used" order for next tab activation when documents close: If this box is checked, the most recently used window becomes the current window when you close another. If this box is unchecked (the default), the tab to the left becomes the current window.
- Dock Window Layouts: Choose which window layout you would like to use as the default. The **Tight Layout** is useful if you will be opening several source files and want plenty of screen space for that. The **Classic Layout** is similar to earlier versions. The **Multi-monitor Layout** allows you to take advantage of multiple screens if you have them.
- **Title Formats:** Choose whether you want filenames in the title areas of windows, tabs, and selector files to be short names, long (full path) name, or relative to the project database.

**User Interface > Lists** The following options can be set from the **User Interface > Lists** category of the **T** 

#### Category

The following options can be set from the **User Interface > Lists** category of the **Tools > Options** dialog:

Display 5 (*) items in the 'Recent Files' list.	
C:\Program Files\SciTools\sample\fastgrep\regexp.h	
C:\Program Files (x86)\SciTools\sample\zlib\examples\zran.c	
C:\Program Files (x86)\SciTools\sample\zlib\trees.c	Remove
C:\Program Files\SciTools\sample\fastgrep\timer.c	
C:\Program Files\SciTools\sample\fastgrep\regsub.c	
Display 5 items in the 'Recent Projects' list.	]
C:\Users\Yvonne\AppData\Roaming\SciTools\sample\fastoren\fastoren udb	
C:\Users\Yyonne\AppData\Roamino\SciTools\sample\mahionoo\mahionoo.ud	Bomovo
C:\Users\Yvonne\AppData\Roaming\SciTools\sample\blow fishvhdl\blow fishv	Remove
	1

- Recent files list: The default is to show five items in a list of recently used files. You can change that default here. You can remove items from the list that you do not want displayed. Note that you can choose File > Recent Files > Clear Menu to clear the history of recent files.
- Recent projects list: The default is to show five items in a list of recently used projects. You can change that default here. You can remove items from the list that you do not want displayed. Note that you can choose File > Recent Projects > Clear Menu to clear the history of recent projects.

.....

User Interface > Alerts Category The following options can be set from the **User Interface > Alerts** category of the **Tools > Options** dialog:

Save On Parse
Always Prompt
$\bigcirc$ Save modified files before parsing
Don't save modified files before parsing
Save On Command
Always Prompt
$\bigcirc$ Save modified files before running a command
Don't save modified files before running a command
Project Close
Prompt before closing the current project
CodeCheck
Prompt when Violation count exceeds 300,000
Entity Filter Window
Sound beep when entity filter entry does not match

These options can be used to re-enable warnings that you have disabled in a warning dialog box.

- Save on parse: Choose what you want done with changed but unsaved source files when the database is to be analyzed. The default is to always prompt you to choose whether to save files. Alternately, you can choose to automatically save changed files or to not save changed files.
- Save on command: Choose what you want done with changed but unsaved source files when a command is to be run. The default is to always prompt you to choose whether to save files. Alternately, you can choose to automatically save changed files or to not save changed files.
- Prompt before closing the current project: If checked (the default), you are asked whether you want to close the current project and all associated windows when you attempt to open a different project.
- Prompt when Violation count exceeds 300,000: If checked (the default), you are asked if you want to continue the CodeCheck when 300,000 violations have been detected.
- Sound beep when entity filter entry does not match: By default, the computer beeps if you type a filter in the Entity Filter that does not match any of the entities of the selected type. You can uncheck this option to turn off these beeps.

.....

User Interface > Windows Category The following options can be set from the **User Interface > Windows** category of the **Tools > Options** dialog:

Editor Windows
Open as:      Captured      Released
Find in Files Search Window
Display 5 🚔 items in the Find in Files 'Find' list.
Directory & Find List: <a>O</a> Global Project
Find in Files Result Window
Restore Default Position
Expand/Collapse code snippet when result double clicked.
Visit result in editor when result double clicked.
Use Contiguous Selection
Graph Windows
Open as:      Captured      Released
Information Browser Windows
Show Code Snippet TooITip in References
Released Windows
Save Released Window Positions
Clear Saved Positions

You can choose various options for different types of windows.

- Editor Windows: Choose whether to open source code files as captured windows contained within the document area of the *Understand* window (MDI windows) or as released windows you can move anywhere on your desktop (SDI windows). The default is captured.
- Find in Files Search Window:
  - **Display:** Choose how many items to list in the drop-down list of recent searches. The default is 5.
  - Directory & Find List: Choose whether lists of recently used search strings should show all searches or only those searches used with the current project.
- Find in Files Result Window:
  - Click **Restore Default Position** if you want to re-dock the results for the Find in Files dialog to the bottom of the *Understand* window.

- **Expand/Collapse code snippet when result double-clicked:** By default, code found in the Find in Files results is expanded to show surrounding lines of code when you double-click. Uncheck this box if you don't want this behavior to occur.
- Visit result in editor when result double-clicked: If you check this box, the code is shown in the Source Editor when you double-click on a result.
- Use alternating row colors: By default, the results for Find in Files and CodeCheck have a slightly darker background for every second row. You can turn off this shading by unchecking this box.
- Use Contiguous Selection: Enable this option if you want the Find in Files results to allow you to select multiple rows while holding down the Ctrl or Shift key. This is useful if you want to copy multiple result rows to the clipboard for pasting into some other document. This option is off by default.
- Graph Windows: Choose whether to open graphical views as captured windows contained within the document area of the Understand window (MDI windows) or as released windows you can move anywhere on your desktop (SDI windows). The default is captured.
- Information Browser Windows:
  - Show Code Snippet ToolTip in References: Choose whether you want several lines of code to be shown in the hover text when you point to a line number shown in the References list in the Information Browser. If the Information Browser does not show line numbers, click the drop-down arrow next to "References" and choose Reference > Full.
- **Released Windows:** Click the **Save Released Window Positions** button if you have released windows from the document area and you want *Understand* to remember the window positions. If you have used this button to save locations, you can use the **Clear Saved Positions** button to forget the locations.

.....

#### User Interface > Application Styles Category

The following options can be set from the **User Interface > Application Styles** category of the **Tools > Options** dialog:

Use Editor Colors	
Item Views	
Default Background	
Default Foreground	
Selection Background	
Selection Foreground	
Rich Text Highlight	
Entity Link	
Unknown Entity	
Tree Row Indentation 12 🚔	
Use alternating row <u>c</u> olors in tables and lists	
Use alternating rows colors in rich text trees.	
Dim highlight color on selected items when the view is inactive	
Selection Background	
Selection Foreground	

You can choose various options for different types of windows.

- Use Editor Colors: Click this button to apply the source editor colors from the Editor > Styles category (page 115) to item views, such as the Information Browser.
- Item View colors: These colors are used in item views, such as the Information Browser. Click a color square next to an item in the list. Use the Select Color dialog to choose a new color for that item.
- **Tree Row Indentation:** You can change the amount of indentation in hierarchical tree displays.
- Use alternating row colors in tables and lists: If checked (off by default), lists and tables, such as the results of a CodeCheck, have shading for alternate rows.
- Use alternating row colors in rich text trees: By default, formatted results have a slightly darker background for every second row. You can turn off this shading by unchecking this box.
- Dim highlight color on selected items when the view is inactive. The Windows default is to dim the highlighting for the selected object when a windows loses focus. If this makes it difficult for you to read the selected object, you can change the behavior by unchecking this box or by changing the background and foreground colors for such items.

# Key BindingsThe functions of keys in Understand can be customized. The Key Bindings category of<br/>the Tools > Options dialog lets you choose how keys will work in Understand:

۶	Understand Options				2	x
	General           General           Ser Interface           Example A Example	Keyboard Sche	eme: Understand	Save As	a Delete	
	Application Codecheck Compare Comparison View	Name	Drimony Soc	By Key Bin	ta Component	<b>P9</b>
		Autocomplete	Esc	uence Allema	Editor	
	Editor Find In Files	Code Check			Application	
	Find Results Graph Window	Collapse	Left	-	Information Browser	
	Information Browser Previewer Window	Collapse All	Ctrl+Shift+L	eft	Information Browser	Ш

- **Keyboard Scheme:** This field allows you to choose groups of keyboard settings that are similar to other applications. The default settings are those native to *Understand*. Other choices are Visual Studio .NET key bindings and the Emacs editor key bindings. If you choose a scheme and click **OK**, that scheme will be used. If you make a change to one of the provided schemes, that becomes a "Custom" scheme. You can click **Save As** to name and save your key binding scheme.
- Search By Name: Type part of a command name and click the M Find icon. All commands that contain that string will be shown.
- Search By Key Binding: Click on the field and press the key sequence you want to search for. Then click the M Find icon. For example, press F3 to find all the key bindings that contain the F3 key.
- **Component:** Different portions of *Understand* have different key behaviors. The "Component" column in the table indicates where a particular command is available. You can see the key bindings for a particular component by selecting a sub-category under the main Key Bindings category in the left side of the dialog. (The Application component applies to dialogs and items not otherwise listed.)

To see a full list of all the current key bindings, choose Help > Key Bindings.

To change the key sequence for an action, follow these steps:

- 1 Use the Component categories or the Search fields to find a command whose key binding you want to change.
- 2 Put your cursor in the **Primary Sequence** or **Alternate** column for the command you want to modify.
- 3 Press the key combination you want to use to perform that action.
- 4 You can't use normal editing keys like Backspace or Delete to edit the keys shown in these fields. To delete the key combination you have entered, click the X in the red circle.

5 When you move focus away from a key binding you changed, you may see a warning message if the key combination you chose is already used. For example:

		🔎 Keyboa	ard Shortcut Conflict	×	
	6 Click Yes to	o make th	The key sequence Ctrl+T is already be Editor : Transpose Line Would you like to override that sequer Yes	eing used by: nce? No hange. Use the Re	estore
	choose one bindings.	e of the pr	ovided Keyboard Schemes to	go back to a defa	ult set of key
Analyze Category	The <b>Analyze</b> c how the projec	ategory o t is analy	of the <b>Tools &gt; Options</b> dialog a zed.	llows you to speci	fy options for
	Pr V	roject Analy ] Show log ] Sound be	vsis Log options g during analysis eep on analysis completion		

- Show standard library files
- Rescan project before analyzing changed files
- Show log during analysis: By default, the Analysis Log is shown while the analysis is being performed. If you uncheck this box, the Analysis Log area is not shown when an analysis is running.
- Sound beep on analysis completion: By default, a beep notifies you when the analysis is complete.
- Show standard library files: For languages whose standard libraries are analyzed by *Understand* (such as Ada), if you check this box the standard library files are shown in the Analysis Log. By default, this box is not checked, and the Analysis Log is shorter.
- Rescan project before analyzing changed files: If you check this box, Understand scans for files that have been added to project directories and to any Visual Studio projects referenced by this Understand project before analyzing the files currently in the project. This has the same effect as using the **Project > Rescan Project Directories** menu command before analyzing the project. By default, this option is off.

**Configure Category** The following options can be set from the **Configure** category of the **Tools > Options** dialog:



Use the New Project Wizard when creating new projects: The check in this box causes the New Project Wizard (page 35) to be used when you choose File > New > Project. If you uncheck this box, you can specify a project database location and filename and then use the full Project Configuration dialog.

Command WindowThe following option can be set from the Command Window category of the Tools >CategoryOptions dialog:

J	Understand Options	2	x
	General	Captured output font	
	Key Bindings	Font: Courier New Size:10 Change Font	]
	Analyze		-
	Configure		
L	Command Window		
	Portability		

This setting controls the font used in the Run a Command dialog to display output from the commands you issue.

# **Portability Category** The Portability category of the **Tools > Options** dialog lets you specify names to use as substitutes for file paths. Named roots are similar to environment variables.

Divident Continues	? ×
<ul> <li>General</li> <li>User Interface</li> <li>Key Bindings</li> <li>Analyze</li> <li>Configure</li> <li>Command Window</li> <li>Portability</li> <li>Dependency</li> <li>Editor</li> <li>Graphs</li> </ul>	Named Roots         Named roots defined here will be use to resolve file paths in projects that use named roots. A named root is simply a mapping of a name to a directory path. Named roots facilitate project portability by allowing directory paths to vary between application users.         Image: Strain
Restore Defaults	OK Cancel Apply

After you have defined a named root, you can use that name in other *Understand* dialogs, such as the Project Configuration, and in "und" command lines (see page 320). This is useful, for example, if you want to share projects with people who reference project files over a network using different paths.

To add a named root, click the **Add Named Root** button. This adds a new row where you can type a name and a path (or click the folder icon to browse for the location).

You can uncheck one or more named roots if you want to temporarily deactivate certain names.

If you change a named root, the project will most likely need to be re-analyzed.

You can define operating system environment variables that will be used as named roots in *Understand*. At the operating system level, define environment variable that have a prefix of "UND\_NAMED\_ROOT\_". The prefix is not used when you reference a named root within *Understand*. For example, suppose you define a system environment variable as follows:

UND\_NAMED\_ROOT\_SOURCEDIR=c:\my\project\dir

The named root that you would then use within Understand is "SOURCEDIR".

If you are using the "und" command-line tool, named roots definitions on the "und" command line have the highest precedence. The next precedence is named roots defined as environment variables at the operating system level, and finally by named roots defined in the *Understand* project configuration.

To use a named root, see Setting File Portability on page 47.

DependencyThe Dependency category of the Tools > Options dialog lets you set options related to<br/>the Dependency Browser (page 140), dependency graphs (page 196), and<br/>dependency exports.

Understand Options	? ×
<ul> <li>General</li> <li>User Interface</li> <li>Key Bindings</li> <li>Analyze</li> <li>Configure</li> <li>Command Window</li> <li>Portability</li> <li>Dependency</li> <li>Sidtor</li> <li>Graphs</li> </ul>	Dependency Analysis "Include" and "Import" references indicate a build-time dependency but don't always indicate a logical dependency (e.g. unused include files). U use Include/Import references Allow PreExpansion Exclude Standard Entities Cytoscape Cytoscape Application Location: Browse

- Use Include/Import References: By default, "includes" and "imports" are treated as dependencies. However, you may want to omit such relationships from dependency lists if they are required for building but are not logically dependent.
- Allow PreExpansion: By default, nodes in the Dependency Browser are expanded. You can uncheck this box to have them closed by default.
- **Exclude Standard Entities:** By default, entities in the standard libraries are excluded from dependency reports, dependency graphs, and the dependency browser. Uncheck this option to include such standard entities.
- **Cytoscape Application Location:** You can browse for the location where you installed Cytoscape (www.cytoscape.org), a free open-source program for analysis and visualization. Specifying this location allows *Understand* to open Cytoscape for viewing the dependency XML files exported as described in *Exporting Dependencies to Cytoscape* on page 244.
Editor Category The following options can be set in the Editor category of the Tools > Options dialog:

Font: Courier New -	Show Indent Guide		
Size: 10 🖨	Insert Spaces Instead of Tabs Indent Width: 10 - Tab Width: 2 -		
Line Endings: CRLF (Windows)	Page Guide		
On Save:	Show Page Guide		
Convert existing line endings	Column: 80		
Convert tabs to spaces			
Add newline at end of file if absent	Whitespace		
Remove trailing whitespace	Invisible		
	Always Visible		
Caret Line	Visible After Indent		
Highlight Caret Line			
	Show End-of-Line		
Externally Modified Files	Margins		
Prompt	Line Number		
Reload	Bookmark		
Reload & Analyze	V Fold		

- **Default style:** Use the **Font** pull-down list to select a font for Source Editor windows. The fonts shown are the fixed-width fonts available on your system. Select a **Size** for the Source Editor text. If you check the **Antialias** box, the font is smoothed. The fields in this area set the default size. You can change it on a per-file basis by choosing one of the **View > Zoom** menu options.
- File Mode: Select the type of Encoding to use when saving source files and the Line Endings character you want used. Many encoding formats are supported. The "System" encoding uses the same encoding format defined for your operating system. You should change these settings only if your other applications have problems opening or displaying files created by *Understand*. By default, these settings apply only to new files you create, including text and CSV files. The previous format is preserved for existing files. However, if you check the Convert existing line endings box, files you save are converted to the format chosen here.

- **Windows** line-endings are terminated with a combination of a carriage return (\r) and a newline (\n), also called CR/LF. When opening a file, a CR, CR, LF sequence is interpreted as a single line ending.
- **Unix** line-endings are terminated with a newline (\n), also referred to as a linefeed (LF).
- **Classic Macintosh** line-endings are terminated with a single carriage return (CR).

If you check the **Convert tabs to spaces** box, tabs are changed to the number of spaces specified in the **Width** field when you save the file. Also, if you check the **Add newline at end of file if absent** box, a new line character is added to a file that doesn't have one when you save the file (checked by default). If you check the **Remove trailing whitespace** box, any spaces or tabs at the end of lines is deleted automatically when a file is saved.

- **Caret Line:** Check the **Highlight Caret Line** box if you want the full line on which your cursor is located to be highlighted.
- Externally Modified Files: If an open file is changed in some other program, Understand detects this. Choose Prompt if you want to be notified and asked to load that changed version. Reload and Reload & Analyze do these actions without prompting.
- Indent: Check the Show Indent Guide box if you want a dotted line to show to column to which lines should be indented.



By default, the **Insert Spaces Instead of Tabs** box is off; turning it on adds spaces to a source file when you press <Tab>.

For **Indent Width**, specify the number of columns in an indentation level. For **Tab Width**, specify the number of columns for each tab stop. For example, if you set the Tab Width to 4, each <Tab> moves 4 columns to the right. If you set Indent Width to 6 and Tab Width to 4, each automatic indentation level is made up of one <Tab> and 2 spaces. You can set a tab width for a specific file to override the project-wide tab width (see page 180). Also, see *Editor > Advanced Category* on page 111 for advanced indentation options.

- Show Page Guide: Check the Page Guide box to display a line similar to the Indent Guide at a defined line width (that is, at the right edge of the code). Set the Column to the character width you want to see indicated.
- Whitespace: Select whether you want to see indicators about whitespace characters. A dot indicates a space, and an arrow indicates a tab. You can choose Invisible (the default), Always Visible, or Visible after Indent. Check the Show End-of-Line box to see the characters that force a line break.

• Margins: Check Line Number (on by default) to turn on line numbering in the source view. Check Bookmark (on by default) if you want bookmarks (red arrows) shown in the margin next to line numbers. Check Fold (on by default) to turn on the ability to "fold" source code entity blocks out of the way.



#### Editor > Advanced Category

You can further customize the code editor's behavior in the Options dialog. To open this dialog, choose **Tools > Options**. Expand the **Editor** category, and select the **Advanced** category.

The following options control how source code looks when you print it from an editor window:

- Font Size: Choose the size of the source code you want to use for printing. To zoom in and out in an individual source code window, see page 183.
- **Color Mode:** Choose a color mode for printing. The choices are as follows. Note that colors other than black and white are printed only if you are using a color printer and the printer driver is set to print in color.

Print				
Font Size: 10				
Color Mode: Normal -				
Wrap Mode: Wrap Word 🔻				
Print absolute file name				
Date: 🔘 Last Modified 🔘 Current				
Date Format: 💿 Long 🔘 Short				
Form feed prints new page				

- "Normal" matches the current display appearance.
- "Invert Light" prints black as white and white as black. This is useful if you set the background to a dark color and the text to light colors for your display.
- "Black on White" prints black code on a white background regardless of the current display appearance.
- "Color on White" prints colored code on a white background regardless of the current display appearance.
- Wrap Mode: Choose the wrap mode you want to use for printing. The default is to wrap words to the next line, but you can choose to truncate lines or wrap at the character level, which breaks words across lines. The line breaks displayed are for printing only; no actual line breaks are added to your source file. See *Line Wrapping* on page 180 to change the wrap mode for screen display.
- **Print absolute file name:** Check this box if you would like the full file path printed at the top of a source file printout, rather than just the filename.
- **Date:** Choose whether to show the date a file was last modified or the current date when printing. The default is the current date.

- **Date Format:** Choose whether to print the date in long or short format. Your system's preferred long and short date format are used.
- Form feed prints new page: If this box is checked, a form feed character in the source code file causes a page break. If you uncheck this box, form feed characters are printed as "FF" and no page break occurs.

The **Copy-and-paste** area lets you control how text is formatted when you copy and paste code into a word processor.

- Include line numbers in rich text pastes line numbers (in bold). HTML is used to format the pasted text. This option is off by default.
- Use preformatted white space pastes code using HTML tags to preserve whitespace. If

Copy-and-paste
Include line numbers in rich text
☑ Use preformatted white space
Auto-complete
Enable Auto-complete
Automatically suggest matches
✓ Ignore case

you disable this option, whitespace is preserved using (non-breaking space) and <br> tags. Some applications may not respect the tag, in which case you can disable this option to force the formatting to match.

The **Auto-complete** options provide for auto-completion of keyword and entities you type in the editor. As you type, words are shown below your text. You can arrow down through the list and press Enter to choose a suggestion.



- Enable Auto-complete: This box is unchecked by default. If you want to enable auto-completion, check this box.
- Automatically suggest matches: If this box is checked, suggestions automatically appear below your typing. If you uncheck this box, you can still see and choose from a list of auto-completion options by pressing Esc while typing.
- **Ignore case:** If this box is checked, suggestions include upper and lowercase versions of the text you are typing.

The **Auto-indent** options allow you to control how tab characters are automatically added to code. If you check the **Enable auto-indent** box, automatic indentation happens as you type in the Source Editor. This smart indenting is currently implemented for C/C++, C#, Java, JavaScript, and Perl code.

- Indent after newline: If this box is checked, when you start a new line, tabs are added so that you begin typing directly below the first character in the previous line. If you uncheck this box, the cursor is always in the first column when you start new lines.
- Tab auto-indents: If this field is set to Never (the default), the <Tab> key always inserts tab or space characters. If it is set to Always, the <Tab> key always adjusts indentation to the "correct" level. If it is set to Leading Whitespace, the <Tab> key causes the appropriate amount to

Auto-indent				
Enable auto-indent				
Indent after newline				
Tab auto-indents: Never				
Trigger characters: :#{}				
V Indent Braces 🔲 Indent Blocks				
Vertical Caret Policy				
💟 Even 💟 Jumps 📄 Strict 📄 Slop				
Slop Value 0				
Unused Entities				
Highlight Unused Entities				
Color:				
Annotation Wrap				
Vrap Annotations				
Column: 60 💂				

indenting in leading whitespace and inserts tabs or spaces everywhere else.

- **Trigger characters:** If you type one of the specified characters, the indentation level for the current line is modified to the correct level based on parsing of the code. For example, a "{" increases the indentation level, and a "}" decreases the indentation level. You can press Ctrl+Z to undo an automatic indentation that just occurred. The default trigger characters are # : { }
- **Indent braces:** If you check this box, the automatic indenting formats code with braces as in the following example:

```
if (true)
 {
    // block of code here
  }
```

• **Indent blocks:** If you check the **Indent braces** box, you can also check this box, which causes the block of code within the braces to be indented further as in the following example:

```
if (true)
{
    // block of code here
}
```

	The Vertical Caret Policy fields let you control how the Source Editor scrolls as the text cursor or current location highlight moves up and down. You can use these fields to optimize the amount of context you see when the Source Editor jumps to a new location. Most users will not need to modify these settings. If you are curious, you can see the descriptions of interactions between these fields at www.scintilla.org/ScintillaDoc.html#SCI_SETYCARETPOLICY.
	• Even: Checking this box causes the source code to scroll the same way both up and down.
	• <b>Jumps:</b> Checking this box causes code to scroll multiple lines as needed to show some context for the current line of code.
	• <b>Strict:</b> Checking this box specifies that you don't want the text cursor to go into the zone defined by the Slop Value. If Slop is unchecked, code scrolls to keep the current line in the middle of the window.
	• <b>Slop:</b> Checking this box lets you define the number of lines at the top and bottom of the Source Editor which you do not want the text cursor to enter.
	• <b>Slop Value:</b> This field lets you set a number of lines at the top and bottom of the Source Editor that the text cursor should avoid.
	The <b>Unused Entities</b> fields let you use a colored background to highlight entities that are never used. By default, this feature is off. If you turn this feature on, the default background is gray for code that defines an unused entity. For example, if a function is never called, all code in that function has a gray background if you enable this feature.
	The <b>Annotation Wrap</b> fields let you cause annotation text to be wrapped at the specified column. This feature is off by default.
Editor > Macros Category	You can record, save, and replay Source Editor macros as described page 180. After you have saved Source Editor macros, you can rename and delete macros using the Options dialog. Follow these steps:
	<ol> <li>Choose Tools &gt; Options, expand the Editor category, and select the Macros category.</li> </ol>
	2 In the top box, choose the macro you want to configure.
	3 Click <b>Edit</b> if you want to rename the macro or assign a different key sequence to trigger it. Note that you cannot edit the actions performed by the macro. To modify the actions, record a new macro.
	4 Click <b>Remove</b> if you want to delete the macro.

# Editor > StylesYou can customize the colors used in the Source Code Editor in the Options dialog. To<br/>open this dialog, choose Tools > Options. Expand the Editor category, and select the<br/>Styles category.

To choose a color scheme with a set of defined colors, choose a scheme from the **Predefined** list. The default scheme is "understand".

To change a color, click a color square next to an item in the list. Use the Select Color dialog to choose a new color for that item.

Predefined: understand  Import Exp	port						
Style	FG	BG	В	1	U	EOL	*
Global							
Default							
Whitespace							=
Number							-
Double-quoted String							
Single-quoted String							
Identifier							
Comment				1			
Keyword			1				
Operator			1				
Preprocessor			1				
Label							
Unclosed Double-quoted String						1	
Unclosed Single-quoted String						1	
Error						1	-

You can change the text foreground (FG) and background (BG) colors for any item. You can also make the text bold (B), italic (I), or underlined (U) for any item. To highlight the whole line for an item, check the EOL box.

You can use the **Import** and **Export** buttons to save your Editor style settings to an Understand Theme (\*.lua) file. This allows you to share styles between computers.

By default, the following color codes are used for the source code:

- Dark blue text: Used for language keywords
- Red text: Used for characters and character strings
- Italic blue text: Used for comments
- Green text: Used for preprocessor statements
- Black text: Used for all other source text and for line numbers
- White background: Used for most source text
- Pink background: Used for inactive lines of code
- Gray background: Used for line numbers
- Yellow background: Used to highlight text in Find Results for Find in Files

Additional items are available for customization depending on your source code language. For example, with C++, you can customize class, enumerator, and namespace names. With Pascal, you can customize the colors of module, routine, and type names. With Fortran, you can customize the colors of block, module, subprogram, and type names. With Ada, you can customize the colors of package names, subprogram names, and type names.

To create additional categories, click **New**. In the User Style dialog, type a name for the style, select the language to which this style applies, and type keywords to be highlighted in this style. Separate the keywords with spaces, line breaks, or tabs. Then click **Save**. You can then set the formatting for your new style.

Editor > Navigation Category You can control the behavior of Browse Mode (see page 168) in the Source Editor. To see this dialog, choose **Tools > Options**. Expand the **Editor** category, and select the **Navigation** category.

Browse Mode				
Activate when Control is pressed				
On Click:				
Action	Modifier			
Edit Source	None			
Vpdate Information Browser	None			
Enable Editor Tooltips				
Number Format: 🔘 Binary 💿 Decimal 🔘 Hexadecimal				

- Activate when Control is pressed: If this is checked (on by default), Source Editor windows use Browse Mode if you hold down the Ctrl key when pointing at an entity.
- Edit Source: If this box is checked (on by default), clicking an entity while in Browse Mode causes focus to jump to the declaration of that entity. You can choose a key (none, Alt, or Shift) that must be pressed along with the click to have this action occur. By default, you must press the Alt key when clicking to jump to the declaration of an entity.
- Update Information Browser: If this box is checked (on by default), clicking an entity while in Browse Mode causes the Information Browser to show information about an entity when you click on it. You can choose a key that must be pressed along with the click to have this action occur. The default is that no key is required along with the click.
- Enable Editor Tooltips: Check this box if you want to see brief information when the mouse cursor hovers over and entity name in source code. The information may include the full name, the type for a variable, and parameters and return values for a function. These tooltips are on by default.
- **Number Format:** Choose whether to display numeric values as decimal, binary, or hexadecimal values in hover text for source code. For example, variables initialized with a numeric literal and enumerated values would show such hover text.

.....

Editor > External Editor Category You can use an editor other than the one provided with *Understand* for viewing and editing your source code. The editor you select is used whenever you open source code. This provides convenient source navigation while using a familiar editor. For example, you can use Microsoft Visual C++ or Emacs as your editor.

You should choose an editor that accepts command line parameters that specify the file to open, and a line and column number to go to.

To change the editor, follow these steps:

- 1 Choose Tools > Options. Expand the Editor category, and select the External Editor category.
- 2 In the Select an External Editor dialog, check the **Use External Editor** box if you do not want to use *Understand* for editing.

Divident Contract Design Participation Providence Provi		? ×
General User Interface User Interface Analyze Configure Command Window Portability Command Window Editor Advanced Styles Navigation External Editor	Use External Editor Editor: Parameters: Use \$file \$line \$col for converted parameters.	

- **3** In the Editor field, click the folder icon and select the executable file for the editor you want to use.
- 4 In the **Parameters** field, type the command line parameters you want to use when opening the editor. Use the \$File, \$Line, and \$Col variables to allow *Understand* to open source files to the correct location.

For example, for the GVIM editor on Unix, the **Editor** is "gvim", and the **Parameters** should be as follows (for GVIM 6.0 or later):

--servername UND --remote +\$line \$file

For the TextPad editor on Windows, the **Editor** is most likely c:\Program Files\textpad4\textpad.exe, and the **Parameters** should be as follows:

\$file(\$line,\$col)

The *Understand* context menus (also called right-click menus) can be made usable in external editors. Steps for doing this are provided in the SciTools blog. For EMACS, vi, and Visual Studio, see scitools.com/support/understand-context-menu-in-ema. For SlickEdit, see scitools.com/support/using-understand-with-an-exter-2.

**Graphs Category** The Graphs category of the **Tools > Options** dialog lets you control options related to how graphs are displayed. These options apply only to certain types of graphs, such as the Cluster Call and Cluster Call Butterfly graphs.

Cluster Graphs	
V Highlight edges on hover.	
Colored cluster backgrounds.	
Animate Transitions	
On node/cluster doubleclick:	Expand/Collapse Clusters Show/Hide Edges In Nodes
Maximum References Shown in Edge C	ontext Menu: 25 📥

- **Highlight edges on hover:** If this option is enabled, relationships (connecting lines) within a dependency graph are highlighted when your mouse cursor hovers over the relationship. This makes it easier to distinguish between overlapping relationships. Text describing the relationship is always shown when your mouse cursor hovers over a relationship; this option does not affect display of the relationship description. See page 196.
- **Colored cluster backgrounds:** By default, the background of a cluster is colored. Uncheck this box to hide such colors.
- Animate Transitions: By default, when you expand or compress a node in a cluster graph, the transition to reorganize the graph and display children of the expanded node is animated. Uncheck this box if you want to omit the animated transition.
- On node/cluster double-click: Controls what happens when you double-click on a node in a graph. By default, clusters are expanded or contracted. You can change this setting to show/hide relationships in one direction or the other. More options let you both expand/contract clusters and show/hide relationships at the same time.
- Maximum References Shown in Edge Context Menu: If you right-click on an line between nodes in a cluster graph, a list of the relationships represented by this edge is provided. By default, up to 25 relationships per node are shown. You can make this limit higher or lower. The maximum allowed value is 99.

The **Graphs** category also lets you customize the display colors, shapes, and arrows for cluster graphs. By default, the settings you make in this area apply only to Architecture Dependency graphs (page 196). If you check the **Use custom style on cluster call graphs** box, your style settings also apply to cluster call graphs (page 272).

Cluster Graph Styles						
In cases where multiple styles apply to a particular node the first matching style is used. For edges, the arrow head and arrow tail are always determined by the first matching style. If the first matching style is custom, then the colors will be a composite of all matching custom styles.						
Style	Fill	Line		Shape	]	
Expandable Files			box3d		New	
Expandable Classes			tab			
Expandable Architecture			box3d		Move Up	
Default Architecture			box		Move Down	
Default Entity			note		move bown	
Edges						
Style	Line	A	rrow Head	ArrowTail	New	
Bidirectional		diam	iond	diamond		
Default		norn	nal	none	Move up	
					Move Down	

To change the color of a type of node or edge (arrow between entities), click the box in the **Fill** column or the **Line** column. To change the shape of the box for an entity or architecture node, use the drop-down list to select a different **Shape**. To change the ends of an arrow, select a different **Arrow Head** or **Arrow Tail**.

If multiple node styles apply, the first matching style in the list is used. You can use the **Move Up** and **Move Down** buttons the change the order of the styles.

You can create custom styles for nodes as follows:

- 1 Click the New button for Nodes.
- 2 Type a unique name for the new style.
- **3** Choose whether this style should apply to only collapsed clusters, only non-clusters, or any cluster.
- 4 Choose whether this style should apply to only architecture nodes, only entity nodes, or all nodes.
- 5 If this is an entity node style, you can choose a way to filter entities that should have this style. Several sample filters are provided, and you can modify the suggested filters to create your own. For example, the "local object" filter applies the style to

locally-defined objects only. The "type ~unnamed" filter applies the style to entities whose type is not unnamed.

🔎 Cluster Graph Node Style - Understand -	(Build 739)
Name (must be unique): User Style	
Collapsed Clusters Non-Clusters	Any Cluster State
Node Type <ul> <li>Architecture Node</li> <li>Entity Node</li> </ul>	Any Node
Filter kindstring (only applies to entity nodes)	OK Cancel

If multiple edge styles apply and the first matching edge style is "Default" or "Bidirectional", then only the first style is shown. However, if the first matching style is a custom style, then multiple edges are drawn for each matching custom style. The arrow head and arrow tail for the first matching custom style are used for all arrows.

You can create custom styles for edges as follows:

- 1 Click the New button for Edges.
- **2** Type a unique name for the new style.
- **3** Choose whether this style should apply to only forward references, bi-directional references, or all references.
- 4 You can choose a way to filter the edges that should have this style. Several sample filters are provided, and you can modify the suggested filters to create your own. For example, the "inherit, inheritby" filter applies the style only to inheritance relationships. The "call ~inactive, callby ~" filter applies the style only to call and callby relationships that are active.

💯 Cluster Graph Edge	Style - Understand - (Bui	ld 739) 🔹 🔁
Name (must be unique):	User Style	
Edge Direction	Bitissetissed	Contraction
Forward	Bidirectional	Any Direction
Filter reference kindstri	ng	
		OK Cancel

# **Analyzing the Code**

Once you configure a project, *Understand* can analyze the project. During analysis, the source files are examined and data is stored in the *Understand* database. After the analysis, the *Understand* database contains lots of data to browse.

When you save or modify the project configuration, a prompt to analyze the project appears automatically. You can also analyze the project in the following ways:

**Project > Analyze Changed Files:** This menu command analyzes all files that have been changed and all files that depend on those changed since the last analysis. This is also referred to as "incremental analysis." To analyze changed files, you can also use the toolbar icon shown here. (Ctrl+R)

•	
Analyze Changed Files	Ctrl+R
Analyze All Files	Ctrl+Alt+R
Rescan Project Directories	

**Project > Analyze All Files:** This menu command forces a full analysis of all project files, whether they have changed since the last analysis or not. (Ctrl+Alt+R)

If some files have been modified but not saved, you are asked whether you want to save all the modified files. You can click **Show Details** to see a list of the files that will be saved.

🔎 Save r	nodified files before Analyzing? - Understand - (Build 738)
?	Would you like to save all the modified files before continuing with Analysis? Don't show this message again. (Press Shift to see message again)
	Yes No Hide Details Cancel
C:\Prog C:\Prog	ram Files\SciTools\sample\getopt\Getopt.java ram Files\SciTools\sample\getopt\LongOpt.java

Analyzing a large project can take some time. If you click **Cancel** while the project is being analyzed you will see a message that says this action will leave the project in an incomplete state. You will need to analyze the project in order to explore it. You can choose to **Abort** the analysis and revert to the previous analysis, **Stop** the analysis and keep the incomplete data, or **Continue** the analysis process.

Summary	
9 of 9 project files analyzed	
Unanalyzed files: 0	
Errors: 0	
Varnings: 13	
Search for missing includes	
Elapsed time: 00:00:00.280	
Completed at: Monday, August 26, 2013 12:10:27 PM	
C++	
C:\Program Files\SciTools\sample\fastgrep\egrep.c	
C:\Program Files\SciTools\sample\fastgrep\regerror.c	
C:\Program Files\SciTools\sample\fastgrep\regexp.c	
🛕 Unable to include file 'stdio.h'	
🛕 Unable to include file 'regmagic.h'	
C:\Program Files\SciTools\sample\fastgrep\regsub.c	

For either command, the *Analysis Log* shows the results as the analysis is being performed.

To save the Analysis Log to a text file, right-click the white background of the Analysis Log and choose **Save As**. Specify the location and name of the file you want to save. Or, you can use **Copy All** to paste the Analysis Log into another application.

can double-click this link to open the Missing Header Files dialog (see page 123).

If you have analyzed the project during this session, you can choose **View > Analysis Log** command to reopen the log. See *Analyze Category* on page 105 for options that affect the project analysis.

You can schedule automatic project analysis. See page 50 for details.

*Tip:* A configured project may be analyzed in batch mode using the command line program *"und"*. Refer to *Using the und Command Line* on page 320 for details on using *"und"*.

Improving theIf your project analysis results in warnings about missing files, choose Project >AnalysisImprove Project Accuracy > Missing Includes and see page 123 for how to use the<br/>Missing Header Files tool.

If your project analysis results in warnings about undefined macros, choose **Project > Improve Project Accuracy > Undefined Macros** and see page 125 for how to use the Undefined Macros tool.

For other types of warnings, revisit the categories of the *Project Configuration Dialog* on page 39 to make sure your project is configured correctly. Multiple similar errors can often be fixed quickly. For advice about tuning configuration settings to improve your project analysis, choose **Project > Improve Project Accuracy > More Information**.

#### Using the Missing Header Files Tool

Configuring your include file directories is important to improving the accuracy of project analysis. If your project analysis results in warnings about missing files, use the Missing Header Files tool as follows:

- 1 Choose **Project > Improve Project Accuracy > Missing Includes** or double-click the link to "Search for missing includes" in the Analysis Log.
- 2 Expand the item for a missing header file and select the source file path to see references to a header file. You will see the code that includes this missing file.

🧏 Missin	g Header Files		- X-
28 Miss	ing Header Files are referenced but not found in the project.		Specify a directory to begin searching for missing
4 🔇	assert.h C:\Program Files\SciTools\sample\zlib\examples\fitblk.c 56		■ Search
C:\Program Files\SciTools\sample\zlib\examples\zpipe.c 15 Source Scitting			
C:\Prog	ram Files (x86)\SciTools\sample\zlib\examples\zpipe.c		
14	<pre>#include <string.h></string.h></pre>		
15	<pre>#include <assert.h></assert.h></pre>		
16 (	#include "zlib.h"	• •	Save Cancel

. . . . . . . . . . . . . . . . . . .

3 Click the ... button in the upper-right corner of the dialog and browse to find a directory that may contain one or more missing header files. All subdirectories of the directory you select will be searched. The search is case-insensitive on Windows.

4 Click **Select Folder** and then click the **Search** button in the Missing Header Files dialog. If any of the missing files are found, the number of files found and their names are listed in the Missing Header Files dialog.

🧏 Missing	Header Files				X
28 Missin  28 Missin  S  S  S  S  S  S  S  S  S  S  S  S  S	g Header Files are referenced but not found in the project assert.h C:\Program Files\SciTools\sample\zlib\examples\fitblk C:\Program Files\SciTools\sample\zlib\examples\zpipe cstdio cstring direct.h errno.h fcntl.h fstream.h	t.	Specify header C:\too Found Select	y a directo r files bls\compiler 1 Missing I Directories # 19	ry to begin searching for missing A
C:\Progra	m Files (x86)\ScTools\sample\zlib\examples\zpipe.c	*	Add A Header	All r Files Four	nd:
15 16 (	<pre>#include <assert.h> #include "zlib.h" </assert.h></pre>	•	S c	stdio:string	
					Save Cancel

- 5 If any of the header files you want to use in the analysis are found, click the + icon next to a directory you want to add or the Add All button below the list.
- 6 The list of missing headers files is updated to show which header files remain missing with a red 🚫 icon and which files have been found with a green 🕖 icon. You can continue searching and adding additional directories as needed.
- 7 Click **Save** to apply your changes to the project configuration.
- 8 Click Yes at the prompt that asks if you want to analyze the project now.

#### Using the Undefined Macros Tool

Configuring your macro definitions is important to improving the accuracy of project analysis. If your project analysis results in warnings about undefined macros, use the Undefined Macros tool as follows:

1 Choose Project > Improve Project Accuracy > Undefined Macros.

. . . . . . . . . . . . . . .

Dundefined Macros							
Undefined Macros:							
Macro	∄ Uses	▼ Ě Files	🕴 Global Value 🎽	Language 🛓			
	≥ ▼	2 -					
DEBUG	6	2		C++			
ERRAVAIL	5	3		C++			
STRCSPN	2	1		C++			
CHARBITS	2	2		C++			
I read	1	1		C++			
-Global Macro Defini	Global Macro Definition						
DEBUG							
Detail View: Directories/Files for DEBUG 🔽 (I) Show Inactive References Save Cancel							

- 2 Select a macro from the list. You can use the headings and fields at the top to sort and filter the list and the **Show Inactive References** box to show or hide such macros. See *Filtering the List* on page 157 for more about using these filter fields.
- 3 Type a definition for the macro in the Global Macro Definition area.
- 4 You can click the **Detail View** button to see the code where the selected macro is used. In this view, you can define a macro value for a specific file or folder instead of project-wide.
- **5** You can select other macros and type definitions for them before saving your changes.
- 6 Click **Save** to apply your changes to the project configuration.
- 7 Click **OK** in the Project Configuration dialog to save the project configuration.
- 8 Click Yes at the prompt that asks if you want to analyze the project now.

# Chapter 4 Exploring Your Codebase

This chapter covers the basic windows in *Understand* and their options in detail. It also covers operations within the Filter Area and the Information Browser.

Details on the use and operation of the Entity Locator and Find in Files for searching for and locating entities are provided in the chapter *Searching Your Source* on page 146.

Details on the use and operation of the **Source Editor** is contained in the chapter *Editing Your Source* on page 165.

This chapter contains the following sections:

Section	Page
PLEASE RIGHT-CLICK	127
Various Windows Explained	128
Entity Filter	129
Information Browser	131
Project Browser	137
Exploring a Hierarchy	139
Dependency Browser	140
Favorites	142

# **PLEASE RIGHT-CLICK**

Sorry for shouting (by using all caps above). In order to make the Understand interface as quick, tight and elegant as possible, we have hidden a lot of power beneath your mouse buttons.

The general rule is that anywhere you look you can right-click to do or learn something.

A second general rule is that right-click reuses windows where it can and Ctrl + rightclick brings up new windows.

Show:	C++ Functions			
Filter:				
adler32	, ,			
adler3:		View Information		Check out all the stuff you can learn or do
allocat		Graphical Views		right-clicking!
allocat		Interactive Reports 🔸		Dight aligh almost
assert		Edit Definition		
atol				anywhere to bring up
bail		Edit Declaration		a menu.
base		Add to Favorites		Ctrl + right-click
bclose		Annotate •		brings up the same menu
BeginC		Remove Annotations		but actions happen
BeginC				in a new window.
bget4		User Tools 🔹 🕨		
bi_flus		Explore •		
bi_reve		Find In		
bi_win				
bits		Metrics Charts		
bits		Browse Metrics		
blast		Filter By Selection		
blen	_			

So please right-click. There will be no more reminders.

# Various Windows Explained...

*Understand*'s GUI has a number of tools for locating and examining entities. This chapter provides a brief list of all these tools and describes the Entity Filter, Information Browser, and Favorites in detail.

The tools available for finding and exploring entities are:

- Entity Filter: Provides an alphabetic list of entities of the selected type. See page 129.
- Information Browser: Provides an explorer for entity characteristics and connections. See page 131.
- Project Browser: Lets you browse a hierarchical file list. See page 137.
- Exploring View: Lets you browse a relationship hierarchy. See page 139.
- Dependency Browser: Lets you browse dependency relationships. See page 140.
- Favorites: Lets you provide quick links to frequently-used entities. See page 142.
- Entity Locator: Lets you filter all entities in a project in complex ways. See page 155.
- Find in Files: Searches multiple files. See page 150.
- Source Editor: Shows source code. See page 165.
- **Contextual Information Sidebar:** Show context information about the current source editor file. See page 164.
- Scope list: Lists the functions or similar constructs in a file. See page 167.
- Architectures: Defines named regions and views of the project. See Chapter 7.
- Graphical Views: Shows connections and structures of entities. See Chapter 10.
- Reports: Generate reports about entities. See Chapter 8.
- Metrics: Generate statistics about entities. See page 226.

# **Entity Filter**

The *Entity Filter* provides a quick list of the selected entity type. You can filter this list to match a text string.

The options in the **Show** list depend upon the languages you have enabled for your project and the types of entities and relationships found in your project. If your project uses multiple languages, the language is listed along with the type.



*Note:* For especially large projects, the All Entities option may be disabled to prevent memory errors.

For each of the entity types, you can quickly find any entity that has been declared (or used) in the source code.

;	▼ C++	Functions (236 of 2080 entities)	98×				
	Show:	C++ Functions	<b>•</b> ] ă				
	Filter:						
	gztell		*				
•	gzunge	etc					
	gzwrite						
in							
1	inf						
1	inflate		Ē				
	in flate_	fast	-				

By default, the entities are sorted in ascending (A to Z) order. You can reverse the order by clicking the drop-down icon and choosing **Sort Descending**.

You can only have one Entity Filter open. If you close the Entity Filters window, reopen it by choosing **View > Entity Filter**.

Using the Filter Field	In the <b>Filter</b> field, you can type a string to match a set of entities. Entity names match if the string is contained anywhere in the name. So, for example, you can type "y" to list only entities that contain a Y or y anywhere in the name.
	By default, filtering is case-insensitive. You can make it case sensitive by clicking the drop-down icon and choosing <b>Filter Case Sensitivity &gt; Case Sensitive</b> .
	If you want to quickly jump to the point in the list where entities begin with a particular letter, just click in the list of entities and type a letter.
	You can select other ways for the <b>Filter</b> field to work. Click the drop-down icon and choose <b>Filter Pattern Syntax</b> . The options are:
	• Fixed String: This is the default behavior.
	• WildCard: With this option selected, you can use * (any characters) and ? (any single character) wildcards for pattern matching. See page 157 for examples.
	• <b>Regular Expression:</b> With this option selected, you can use Unix-style regular expressions. See page 157 for an overview.
	To see only unknown or unresolved entities, click the drop-down icon and choose <b>Filter</b> <b>Unresolved Entities &gt; Hide Resolved Entities</b> . To see only resolved entities, click the drop-down icon and choose <b>Filter Unresolved Entities &gt; Hide Unresolved Entities</b> .
	When you are finished using a filter and want to see all the entities for the selected type, click the drop-down icon and choose <b>Clear Filter</b> .
	If you change the type of entity in the <b>Show</b> field, any filter you have typed is cleared if the <b>Clear Filter Text on Filter Type Changes</b> option is selected in the menu available from the drop-down icon <b>E</b> .
	You can select from filters you have used by right-clicking the Filter area and choosing from the <b>Most Recent Filters</b> list. Filters are shown in this list if you have selected any entity found using a filter.
Customizing the	You can modify how the Entity Filter lists entities as follows:
Display	By default, the full entity name is shown in the Entity Filters list and entities are alphabetized by their full name. This name may include a class prefix or other language-specific prefix type. To list entities by their "short", unprefixed names, click the drop-down icon and choose <b>Entity Name as &gt; Short Name</b> .
	By default, only the name of the file is shown in a Files list in the Entity Filter. This name does not include the file location. To list files including their locations, click the drop-down icon and choose <b>File Name as &gt; Relative Name</b> or <b>File Name as &gt; Long Name</b> .
	By default, only the name of a function or method is shown. To also show the parameters of such entities, click the drop-down icon and choose <b>Show Parameters</b> as > Full Parameters or Show Parameters as > Short Parameters.

Root Filters
Notice that there are the filter type names that contain "Root", as in Root Calls, Root Callbys, and Root IncludeBys. These "Root" types show only the top of a given tree. The tops (or bottoms) of relationship trees are often helpful points to begin exploring code that is new to you.
Root Calls: Lists only entities that call others, but are not called themselves. These are either high-level code (mains), code called by hardware (interrupt handlers), or dead (unused) code.
Root CallBys: Lists only entities that are called by others, but that do not call anybody else. These are low-level routines.
Root IncludeBys: Lists only files included by others, but not included themselves. These are "lower" level include files.
Root Classes: Lists only classes not derived from other classes. These are candidates for lower level classes, or library classes.

- Root Decls: Lists only the highest level declaring routines. (Ada)
- **Root Withs:** Lists only program units (packages, tasks, subprograms) that With other program units, but are not withed by anybody else. (Ada)

#### **Information Browser**

When you click on an item in the *Entity Filter* or in a number of other windows, the *Information Browser* updates to show everything that *Understand* knows about that entity. The *Information Browser* shows this data as a tree whose branches can be expanded individually or all at once.

If the Information Browser isn't open, you can open it by either clicking on an item in the Entity Filter or Project Browser. You can also right-click on an item anywhere and choose **View Information**. Or, choose **View > Information Browser** from the menus.



Everything *Understand* knows about an entity can be learned using the *Information Browser*. The information is shown in a tree. The tree can be expanded selectively or in bulk. Each terminating item (leaf) of a tree provides some information about that entity.

All information in an *Information Browser* window can be saved to a text file, or copied and pasted via standard Windows or X11 copying functions.

As you drill down you can change which entity you are learning about. Each time you change the entity, it is remembered in the Information Browser history for quick backtracking.



Drilling Down a Relationship

Drilling down the tree works as expected (mostly). To expand a tree, click on the + sign. To close the tree click on the - sign.

Þ

Right-clicking brings up a menu that includes expand/collapse options. **Expand All** provides a shortcut to expand all levels of the selected branch.

To open or close the entire tree, rightclick on the top item and choose **Expand All** or **Collapse All**.

See Saving and Printing Information Browser Text on page 136 for details on the other options in this context menu.

Metric	Allow Pre-expansion		
	Expand	Right	
	Expand All	Ctrl+Shift+Right	
	Collapse	Left	
	Collapse All	Ctrl+Shift+Left	
	Сору	Ctrl+C	
	Copy All	Ctrl+A, Ctrl+C	

Displaying More or Less Information	If you click the relation next to a bold heading such as <b>Calls</b> , <b>Called By</b> or <b>References</b> in the Information Browser (or right-click on the heading), you'll see options that let you modify how that entity is listed. These options include:				
	• Fullname: If checked, the fully-qualified name of the entity is shown.				
	Parameter: Lists the parameters.				
	• Reference: Choose "Full" to include the file and line location of the reference.				
	Return Type: Lists the return type.				
	• Sort: Controls the sort order of the list.				
	• <b>Type:</b> If checked, the datatype is shown.				
	• <b>Filename:</b> Controls whether the reference format is short, long, or relative to the project database.				
	<ul> <li>View by: For lists of references, you can choose whether to display a flat list of references or to group references by the files that contain them or by one of the defined architectures.</li> </ul>				
	• <b>Group by:</b> For C++ classes, you can choose whether to sort class members by the type of access available (public or private) or the kind of member (function or object).				
Searching the Information Browser	If you click the binocular icon at the top of the Information Browser (or click in the Information Browser and press Ctrl+F), a Find bar appears at the bottom of the Information Browser.				
	Type text in the box and click a forward or backward arrow to find an occurrence of the string in the Information Browser text. All text is searched, including node names and items that are currently hidden by collapsed nodes. If you type a string that does not appear anywhere in the Information Browser text, the field turns red.				
	Last Analysis: 8/26/2013 12:10:27 PM Case Sensitive Whole Words				

To make the search **Case Sensitive** or to match only **Whole Words**, use the dropdown arrow to select those commands.

Syncing the	You can have multiple Information Browser windows open if you uncheck the <b>Sync</b> box.
Information Browser	Selecting an entity or choosing <b>View Information</b> updates the Information Browser that has its <b>Sync</b> box checked.
	The <b>File Sync</b> box synchronizes the Information Browser with the file in the active

The **File Sync** box synchronizes the Information Browser with the file in the active Source Editor.



#### Visiting Source Code

In general, if you double-click on an entity in an informational window (*Information Browser* or *Entity Filter*) the declaration of that entity will be loaded into the *Document Area*.



Another way to visit source from any entity you see in *Understand*, is the context menu. Where appropriate, an entity's context menu contains an **Edit Source** (Ctrl+Shift+S) menu item. In some cases, there are separate menus items for **Edit Definition** and **Edit Declaration** (Ctrol+Shift+D) or separate menus for other language-specific locations. If you have a \*.c or \*.h file selected, the **Edit Companion File** command opens the other file if one exists for that filename.



Left-click on any reference to visit that location in the source code.

Right-click on the "References" title for the node or click the down-arrow next to the node to choose how to organize the references. Your choices are the default flat list, and all of your architectures.



# **Viewing Metrics** The last node on the Information Browser tree is **Metrics**. This branch shows the metrics available for the current entity.

By default, when you switch to another entity in the Information Browser, the Metrics node is closed automatically. This is because it can take a long time to update the metrics for each entity in a large project.

If your project is small enough that updating metrics as you switch between entities does not take a long time, you can right-click on the **Metrics** node and choose **Allow Pre-expansion**. The Metrics node will then stay open when you change entities. You see the following warning about the time required for metric updates.



See Metrics Reports on page 226 for details on metrics.

Saving and Printing Information Browser Text	All text shown in the <i>Information Browser</i> can be copied to the clipboard for pasting into another application as unformatted text. Only the currently expanded branches are pasted. When saving or pasting in text format, the branches of the tree are represented by indents in the text.			
	The context menu offers choices to <b>Copy</b> (only the selected line) and <b>Copy All</b> . For entities with a class path and files, the <b>Copy</b> command copies the short name and the <b>Copy Full Name</b> command copies the full class path or file path.			
Entity History	As you explore your code, you can go many places quickly. Often you want to backtrack to explore a new path. To help you do this, the <i>Information Browser</i> contains a full history of what is has displayed. The <i>Information Browser</i> history can be found in the upper-left corner:			
	Click small arrows to see a full history list Choose from menu to jump to that point in your exploration.			
	History arrows. Click to move back and forth in history. History arrows. Click to move back in history. History arrows. Click to move back and forth in history. History arrows. Click to move back and forth click to move Defined in: try.c ♥ Defined in: try.c ♥			
	Use the right and left arrows to move back and forward in the history list. The down- arrows show the whole list.			
	You can choose <b>Clear History</b> from the drop-down history list to clear the browsing			

You can choose **Clear History** from the drop-down history list to clear the browsing history.

# **Project Browser**



To open the Project Browser, choose **View > Project Browser** from the menus.

By default, the Project Browser is in the same area as the Entity Filter (and the Architecture Browser). Use the tabs on the left to switch between browser tools in this area.

The Project Browser shows the project files in their directory hierarchy. You can expand and collapse the tree as needed.

Press Ctrl+F to display a search line at the bottom of the Project Browser.

The **File Sync** box synchronizes the Project Browser with the file in the active Source Editor.

The context menus when you click on a file in this view offer the same commands as other views such as the Information Browser or Entity Filer.

The context menu when you click the background of this view offers a number of options. The options with icons are also available in the toolbar for the Project Browser.

Add a file to the project		Add Existing File	Ctrl+Shift+F
Remove selected file from project		Remove	Ctrl+Shift+Del
Open file in Source Editor	2	Edit	Ctrl+Shift+E
Open file with default OS tool	Ø	Open Externally	Ctrl+Shift+O
Copy filename to clipboard		Copy Filename	
Search for text in Project Browser	抖	Incremental Find	Ctrl+F
Search in reverse direction		Incremental Find Previous	Ctrl+Shift+F
Copy selected file to clipboard		Сору	Ctrl+C
Change sort order of files		Sort By	+

For a file, the context menu includes additional commands, including commands to open graphical views, rename the file, analyze this file only, find uses of the filename in project files, and add the file to the Favorites list.

The **Add Existing File** command lets you browse for and add source code files to the project.

To use the **Remove** command, select one or more files and folders and choose this command. The Confirm Project Modification dialog lists files that will be deleted from the project if you click **Yes**.

The **Open Externally** command opens an operating system dependent tool for the directory or file. For example, on Windows it opens a directory using the Windows Explorer. For a file it opens the default tool for the file extension.

The **Incremental Find** command opens a Find bar at the bottom of the Information Browser. Type text in the box and click an arrow to find an occurrence of the string in the Project Browser text. All text is searched, including files in folders that are currently closed. If you type a string that does not appear anywhere in the Project Browser, the field turns red. See page 133 for details on search options.

The **Sort By** command lets you organize the list alphabetically by filename or by file extension.

# **Exploring a Hierarchy**

The Exploring view lets you browse up and down a relationship hierarchy within your project.

<ul> <li>Explorin</li> </ul>	ig main calls						98×
Explore:	init_block		74277				
main	C_Fwrite deflateEnd deflateInit_ deflateReset	deflateInit2_	deflateEnd deflateReset zalloc zcalloc zcfree	* * *	_tr_init adler32 crc32 lm_init	+	init_block tr_static_
•		III					*
Referenc	es: _tr_init calls init_block - trees.c(405)				Navigation Genera Jump to	Actior te Syn First I	ns Ics Reference

The context menu in the Information Browser, Entity Filter, and Project Browser offers commands to Explore certain types of entities. The command will be similar to **Explore** > **Explore Called By/Calls** or **Explore > Explore Includes**.

If you click on an item in one column, you see its relationships in the columns on either side. As you choose items to the left or right, columns resize to show more of the hierarchy. Calls and Includes go from left to right. Callbys and Includebys go from right to left.

If you double-click an item, a Source Editor window shows the entity's definition.

The **References** area shows the line number of the currently highlighted relationship. Double-click to visit that code.

If you check the **Generate Syncs** box, then the Information Browser automatically displays information about any entity you select in the Exploring window. Holding the Shift key down temporarily activates this behavior.

If you check the **Jump to First Reference** box, then the Source Editor automatically displays the initial reference to any entity you select in the Exploring window. Holding the Ctrl key down temporarily activates this behavior.

Click the drop-down icon 👔 if you want to enable any of the following display options:

- Show Long Name: Shows the long name of each call.
- Show Parameters: Shows the parameter list for each call.

### **Dependency Browser**

The Dependency Browser lets you examine which items are dependent on others. You can use the Dependency Browser with architecture nodes, files, classes, packages, and interfaces.



To open the Dependency Browser, right-click on an architecture node, filename, class name, or package name anywhere in *Understand*, for example in the Entity Filter, Information Browser, or a graphical view. Choose **View Dependencies** from the context menu. Or, click the **View Dependencies** button in a dependency graph. You can also open the Dependency Browser by choosing **View > Dependency Browser**.

The left panel shows the item you selected and the items it contains. The right panel shows items that either depend on the item selected in the left panel or are dependent on that item,

ependency Kind:	Depends On	Ŧ	
	Depends On		
	Depended On By		1

depending on your choice in the **Dependency Kind** field. For example, an item depends on another if it includes, calls, sets, uses, casts, or refers to that item.

You can expand hierarchies in the left and right panels. For example, when you view dependencies for an architecture node, you can expand it to see lower-level architecture nodes, then files, then the entities in the files. Letters next to items identify whether they are architecture nodes ("a"), files ("f"), classes ("c") or entities in files ("e"). You can also right-click on either panel and choose **Expand All** or **Collapse All**.



D

You can use the **Group By** field to select an architecture to control how the items in the right panel are organized.

In the **Files, Classes, Entities** drop-down list, you can select whether to show each of these types of items in the right panel. If multiple items are checked, files are listed below the lowestlevel architecture node that applies, classes are listed below files, and entities are listed below classes.

In the **All Dependencies** drop-down list, you can select types of dependencies to show. By default, all types are shown.

If you have the **Sync** box checked in the Dependency Browser and click on a relationship (connecting line or "edge") in a dependency graph, the graph's **Show** and **Group By** settings are copied to the Dependency Browser and the **Dependency Kind** is changed to Depends On.

Click the A Graph icon to create a graphical view of the dependencies currently shown in the Dependency Browser. While the Dependency Browser shows one level of dependency, graphical views can show multiple levels. See page 196 for details.

File	s,Classes,Entitie 🔻	
V	Files	f
V	Classes	l
V	Entities	l
_		
		h



Click the **Export** icon to export a comma-separated values (CSV) report of dependencies for the top item in the left panel of the Dependency Browser. You can also use the mouse to select items in the right panel of the Dependency Browser. Then right-click on a letter icon and choose **Copy** to place the currently selected text on your clipboard for pasting into other applications. See page 242 for other ways to export information about dependencies.

Click the **Tavorites** icon to add the current dependency to your Favorites list. See page 142.

If the **Reuse** box is unchecked, a new Dependency Browser is opened when **View Dependencies** is selected. The Reuse box is checked by default.

If the **Sync** box is checked, the Dependency Browser displays information about any architecture node, file, class, or package you select in the Project Browser, Entity Filter, Architecture Browser, or similar window.

If you have a dependency graph open and the **Sync** box is checked, the Dependency Browser syncs to show any relationship you select in the dependency graph, and the two nodes connected by the relationship are highlighted in the Dependency Browser. In addition, **Show** and **Group By** settings from the dependency graph are synced with the Dependency Browser and the **Dependency Kind** is changed to Depends On.

Click the drop-down icon 👔 if you want to change any of the following display options:

- Architecture Name. The default is to show the relative name, but you can select short name or long name instead.
- Entity Name. The default is the short name. You can select the long name instead.
- File Name. The default is the short name. You can select the relative name or long name instead.

- Use Include/Import References. By default, "includes" and "imports" are treated as dependencies. You may want to omit such relationships if they are required for building but are not logically dependent. This option can also be controlled in the Dependency category of the **Tools > Options** dialog (page 108).
- Allow Pre-Expansion. If you enable this option, nodes in the Dependency Browser are automatically shown as expanded. This option is on by default, and can also be controlled in the Dependency category of the **Tools > Options** dialog.

### **Favorites**

You can mark all kinds of things in Understand as "Favorites" so that you can quickly access them as you would web pages in a browser's Favorites group. Your favorites can include entities, code locations, graphs, Information Browser displays, and dependencies. You can also store multiple plain text strings in your favorites, so that you can quickly copy one of your saved strings to the clipboard.

Favorites are saved as part of a project. If you want to mark code locations on a crossproject basis, see *Annotations* on page 184.

.....

**Creating a Favorite** To mark an entity as a favorite, follow these steps:

Entity

1 Select an entity name and right-click in source code, the Entity Filters area, the Information Browser, a graphical view, or anywhere else entities occur.



- 2 Choose Add to Favorites > Add To New Favorites (or an existing favorites group). This adds a link to the entity itself, even if the line number changes later. If you've already created a favorites group, you can select it from the submenu instead of using Add To New Favorites.
- Alternately, if you right-click on a source file, you can choose Add Location to Favorites to add the line number in the file to a favorites group.

See *Creating a Plain Text Favorite* on page 145 for information about the **Add Selection to Favorites** command, which stores text for pasting from the clipboard.

4 If you choose to create a new favorites group, you see the New Favorite dialog. Type a name for the new group and click OK.



5 When you create a favorite, the Favorites group opens.

 Creating a Favorite
 Besides entities and code locations, favorites can include graphical views, Information Browser views, and Dependency Browser views.

 To add a favorite for any of these items, click the review. By default, the view is added to the last favorites group you used. If you want to place it in a different group, choose a group from the drop-down menu.

 Using a Favorites
 To open a Favorites group, choose View > Favorites and choose a group name from the menus. You see the favorites saved to that group.

 Parser Favorites
 Favorites



In the Favorites view, you can use the drop-down list to switch to a different Favorites list.

Click on a link in the favorites group to jump to that location. You can open all the favorites in the current list by clicking the **Open Favorites** icon and close all the favorites in the list by clicking the **Close Favorites** icon.

You can add all the currently open files and graphical views to a Favorites list by clicking the small dimple icon in the upper-right corner of the document area and selecting the **Add Open Editors/Graphs to Favorites** command.



As with just about every place in Understand, you can right-click on a favorite to see a context menu that includes commands such as View Information, Graphical Views, and Find In.

An icon to the left of each favorite (and the text after the name of the favorite) identifies each favorite's type. For example, in the previous figure the first five favorites link to various types of entities or line numbers in the code.

Favorites with a 🛸 file icon link to a file.

Favorites with an (1) information icon link to an Information Browser view.

Favorites with a 🚠 dependency icon link to a Dependency Browser view.

Favorites with a 🛃 graph icon link to a graphical view.

Favorites with a Plain Text Favorite on page 145 to store text as a favorite. When you click on a text favorite, the text is placed in your clipboard and you can paste it into Source Editor windows or other applications.

If you click the S **Configure** wrench icon at the top of a Favorites group, additional toolbar icons are displayed. These let you manage favorites you have already created as follows:

 $\land$   $\checkmark$  The arrow icons move the selected favorite up or down in the group.

is Click this icon to create a header that you can use to organize your Favorites.

Click this icon to create a text favorite. See *Creating a Plain Text Favorite* on page 145 for details.

Delete the selected favorite from the group. You can also delete a favorite by going to its location, right-clicking, and choosing the **Remove from Favorites** command.

Rename the current favorites group.

Delete the current group of favorites.
#### Creating a Plain Text Favorite

You can store multiple plain text strings in your favorites, so that you can quickly copy a saved string to your clipboard and paste it as needed into your code. For example, you might use a standard comment at the beginning of files or elsewhere in your code.

To save text as a favorite, follow these steps:

1 Select text in a Source Editor and right-click.



- 2 Choose Add Selection to Favorites and select a favorites group from the submenu.
- **3** When you create a favorite, the Favorites group opens.

You can also create a text favorite by clicking the 📓 icon in the Favorites area (which you can see if you select the 🔊 **Configure** icon at the top of the Favorites area). You see the New Text Favorite dialog.

P New Text Favorite
New Text Favorite Name
Please specify the name of the new Text Favorite item to save.
this
Add any text that you would like to save with this Favorite item.
ThisIsAReallyLongClassName
All Text Favorites copy to Understand Editor also. OK Cancel

In the first field, type a short name to be shown in the Favorites list.

In the second field, type the full text of the favorite.

To use a text favorite, double-click on the name of the favorite in your Favorites list. This copies the longer text from the second field to your clipboard, so that you can paste it into a Source Editor.

If you check the **All Text Favorites copy to Understand Editor also** box, then when you click on a Text Favorite, the text you typed in the second field is automatically pasted into your current Source Editor window at the text cursor position.

# Chapter 5 Searching Your Source

This chapter covers how to use *Understand*'s Find in Files and Entity Locator features to locate thing in your source code.

This chapter contains the following sections:

Section	Page
Searching: An Overview	147
Instant Search	148
Find in Files	150
Entity Locator	155
Finding Windows	160
Searching in a File	163

# Searching: An Overview

Finding things in large bodies of source code can be difficult, tedious, and error prone.

*Understand* offers a number of ways to search for strings in your source code or to locate particular lines. The commands for these options are located in the **Search** menu. These commands are described in the locations listed in the following table:

Search Command	See	Comments
Entity Locator	page 155	project-wide, entity-based
Find in Files	page 150	project-wide, text-based
Replace in Files	page 153	project-wide text-based
Instant Search	page 148	project-wide text-based
Find Next and Previous	page 163	single file
Find & Replace	page 163	single file
History	page 161	project-wide
Bracket Matching	page 178	single file
Favorites	page 142	project-wide
Contextual Information Sidebar	page 164	single file
Bookmarks	page 181	project-wide

Each of these searching methods has advantages and disadvantages. Together they provide powerful ways to easily find what you need to find to better understand and modify your code.

See page 128 for a more complete list of the code exploration tools in Understand.

## **Instant Search**

Instant Search lets you search your entire project instantly, even if it contains millions of lines of source code. As you type, you can see terms that match the string you have typed so far.

The search box for Instant Search is in the upper-right corner of the *Understand* window.

If you don't see this field, choose View > Toolbars > Search from the menus.

To begin searching, click in the Search field and type a string you want to find. You can also press Ctrl+Alt+S or choose **Search > Instant Search** from the menus to move your cursor to the Search field.



The easiest way to use Instant Search is to type a string that you want to match in your code. Press Enter after typing a search string to see a list of files that match your search in the Search Results area.

Right-click in this area to **Expand** or **Collapse** the results tree. Choose **Find** to use the **Previous** and **Next** icons to move through the results one-by-one. You can double-click on a file to open the file and see the line of code in the Search Results area.

👻 Search Resu	ilts for 'bufsiz' (2)	ଡୁ ୫ ×
C:\Program File 130	es\SciTools\sample\fastgrep\timer.c char dbuf[ <mark>BUFSI2</mark> ];	
▷ C:\Program Files\SciTools\sample\fastgrep\try.c		
		.::

A number of more powerful search options are supported with Instant Search. The syntax used by this field is based on the syntax used by Apache Lucene, an open-source text search engine library. See

lucene.apache.org/core/3\_6\_2/queryparsersyntax.html for syntax details.

The following list explains some of the syntax options available:

- Searching is case insensitive. A search for test also matches "Test" and "TEST".
- Unless you use wildcards, searching matches whole words. A search for test does not match occurrences of "testfile".
- The wildcards available are \* (any number of letters and digits), ? (any single letter or digit). You cannot use a wildcard as the first character in a search string.

• When indexing the code (which happens in the background), Instant Search breaks code into searchable strings by splitting the code at white space and punctuation (and syntax conventions for C/C++, Java, and Ada). So, the searchable strings in the following line of code are "foreach", 1, and 10:

foreach (i=1, i<10, i++)</pre>

- You cannot use Instant Search to find strings that cross punctuation boundaries or to search for punctuation itself. For example, you cannot search for "i=1". You can search for strings that contain spaces (such as text in comments) by surrounding them with quotes.
- You can narrow the search to look within strings, identifiers, and comments. By default, it searches for all three types of matches. For example, the following search finds "test" only in quoted strings:

string:test

The following search finds "test" only in identifiers such as variable and function names:

identifier:test

The following search finds "test" only in comments:

comment:test

- You can use Boolean searches. The default is that multiple search terms are ORed. So, a search of "for delta" is the same as a search of "for OR delta". Both match files that contain either "for" or "delta". Remember that the search string is used to match terms in the entire file, not just in a single statement.
- If you want to AND the terms, use a search like "for AND delta". This matches files that contain both "for" and "delta".
- You can use the + operator to require that a search term exists in all documents found. For example, the following search finds documents that all contain "delta" and may contain "for":

+delta for

You can use the NOT (or -) operator to remove any documents that contain a
particular search term from the results. For example, the following searches find
documents that contain "delta" or "delta0" but not "delta2":

```
delta delta0 NOT delta2
delta delta0 -delta2
```

 You can use parentheses to define the order of Boolean operators in searches. For example:

```
(delta0 OR delta1) AND change
```

You can perform a "fuzzy" search by placing a tilde (~) at the end of a search term.
 For example boo~ matches foo, too, and book.

# **Find in Files**

You may search all project files or another selection of files for the occurrence of a text string or regular expression. Matches are shown in the *Find Results* window and can be visited in the source code by double-clicking on any line in the results. You can switch between the Find in Files and Replace in Files (see page 153) dialogs by checking the **Replace** box.

To open the Find in Files tool, choose **Search > Find in Files** from the menu bar, choose **Find in...** from any context menu, or press F5.

	✓ Find In Files	×
	Find: error	-
	File Types: *.* 💌	1
	Case Sensitive Match Whole Words	
	Search Type: Fixed String	•
	Find In: Project Files	-
✓ Find Results	Semantic Options	 @a×
🔯 🙌 💥 🚺 🥟 📀 C:\Program Files (x86)\SciTools\s	Only Show Results In:	Ē
87 ▲ Search for error: 87 results found in 13 files of 33 files s 1 ▷ deflate.c: 1245 z_error("invalid	Strings	*.c,*.h; Cas ▲
<pre>1 a example.c: 24 fprintf(stderr, "%</pre>	Statements	
22 #define CHECK_ERR(err, msg 23 if (err != Z_OK) { \ 24 fprintf(stderr, "\$	Inactive Code	
25 exit(1); \ 26 } \	Stop Find	
<pre>1</pre>		
< III		+

The Find in Files area allows you to search multiple files for the occurrence of a string. In previous versions, this feature was called Hyper Grep for its similarity to the Unix command *grep*. Specify a search as follows:

- **Find:** Type the string for which you want to search. The other fields control how this search is performed. The drop-down list lets you select from recent Find strings.
- File Types: You can select file extensions for various languages to narrow the search. Or, type your own file extension pattern. Leave this field blank to search all files. You cannot use this field if you have the Find In field set to "Open Files". Check the "!" box to the right of the File Types field to exclude the selected file types from the search and search all other files in the project.

- **Case Sensitive:** Check this box for case-sensitive searching. The default is to ignore case.
- **Match Whole Words:** Check this box to match whole words only in regular expressions ("test" matches "test" but not "testing"). For fixed string and wildcard searches, word boundaries are ignored.
- Search Type: Choose whether to use Fixed String, Wildcard, or Regular Expression matching. See page 158 for details.
- Find In: Choose whether to search project files (either all files or just the open files), files in architecture nodes you select, files in directories you select, or files you select.



For Architecture, Directory List, and File List searches, click + to add a location. Click the pencil icon to modify the selected

location. Click the red X icon to delete the selected location. You can uncheck a location to temporarily disable searching it.

If you select **Architecture**, click + to browse for an architecture node.

If you select **Directory List**, click + to browse for directories. You can click the minus icon to exclude the selected directory from the search. Sort the list with up and down arrows.

If you select **File List**, you can click + to browse for files.

If you select **Open Files**, all files that are currently open are searched. The **File Types** specification can be used to limit which open files are searched.

When you right-click on an entity in source code or elsewhere, the **Find In** command lets you choose one of these options for the selected text string. The Find and Find In fields are filled in for you automatically.

- Semantic Options: If you choose to Find In "Project Files" or "Architecture", you can check the Only Show Results In box to be able to control which matches are reported. Then you can check any combination of the Comments, Strings, Statements, and Inactive Code boxes to include those types of lines in the results. You must check at least one of these boxes if you check the Only Show Results In box.
- Replace: Switch to the Replace in Files (see page 153) dialog by checking this box.

Click **Find** after specifying the search criteria. A list of all matching instances will be displayed in the *Find Results* window. If the search is taking a long time and you want to change the criteria, you can click **Stop**.

Find Results

The Find Results window lists the matches found. Each line where the string occurs is listed in the Results list.



You can view the source code for a match by double-clicking on a result. This opens the Source Editor and highlights the match. See *User Interface > Windows Category* on page 101 for ways to customize the Find Results display.

Multiple searches are shown in the results list. Right-click on the background of the window and choose **Expand All** to expand all search nodes in the window. Or, choose **Collapse All** to compress the list to just the top-level search listing.

The toolbar (and context menu) for the results provides the following controls:

- Q Go to Find in Files dialog.
- M Search within the currently selected set of results (using the same search bar described in *Searching the Information Browser* on page 133).
- 💓 Delete the current set of results.
- 🚺 Delete all the results.
- 🧭 Open the selected match in the Source Editor.
- O Move to the previous or next match.
- Check the Criteria box to show the settings used to perform the search.

The drop-down icon in the upper-right corner of the Find Results area provides the following commands:

- By default, only the names of files are shown. To show full file paths, select "Long Names" from the **Display Files As** drop-down.
- Use the **Organize Results By** drop-down to change the organization of the most recent results. The choices are a flat list (the default), a file-based list, and hierarchies using the architectures.
- Choose **Show Criteria** to have the line that shows the number of results also list the search options that were used for each search.

	<ul> <li>Choose Expand All by Default to automatically expand all lines when new results are added.</li> <li>Choose Clear Results Before Search to automatically clear the results of the previous search whenever you run a new search.</li> <li>From the context menu, you can choose Copy or Copy All to copy the contents of the window as text for pasting elsewhere.</li> </ul>		
	You can reopen the Find Results window <b>Results</b> . All the results from the current toolbar in the Find Results window to de	w by choosing <b>Search &gt; Show Find in Files</b> session are shown unless you have used the elete some results.	
Replace in Files	You can use the Replace in Files tool by choosing <b>Search &gt; Replace in</b>	✓ Replace In Files	
	<b>Files</b> from the menu bar or by checking the <b>Replace</b> box in the Find in Files tool	Find: interpilate -	
	The fields in this tool are the same as those in the Find in Files tool with the following exceptions:	Replace.     Interpolate       File Types:     *.*       V Case Sensitive     Match Whole Words	
	<ul> <li>There is a <b>Replace</b> field where you type the text you want to replace the matched string.</li> </ul>	Search Type: Fixed String   Find In: Project Files	
	<ul> <li>In the Search Type field, you can select "Regular Expression - Multiline," which lets your regular expressions match strings that cross line boundaries.</li> </ul>	Semantic Options          Image: Semantic Options         Image: Only Show Results In:         Image: Option O	
	To switch between the Replace in Files and Find in Files tools, check or uncheck the <b>Replace</b> box just above the Stop button	Statements Inactive Code Replace	
	Understand checks for any unsaved source files. If there are unsaved files, you must click <b>Yes</b> to save all unsaved changes before making or	Stop A Find	
	If you click <b>Replace All</b> , you are asked The changes will be saved automaticall the changes.	if you want to replace all results automatically. y, so you should be sure you want to make all	

If you click **Preview Replace**, you see the Preview Replace Changes window. You can use this window to accept or reject replacements on a change-by-change basis, file-by-file basis, or all at once.



The top area shows the pre-change code on the left and the post-change code on the right. Replacements are in pink and the currently selected replacement is highlighted in blue. The left side has the **Hide Common Lines** option set so that most lines that will not be affected by the replacement are hidden.

The middle area shows the replacements in patch file format. Such patch files can be used with the Unix patch tool and other similar programs. You can hide this area by clicking the small **fold** icon above the area.

The lower area lists the files where replacements will be made and the number of replacements accepted and unresolved.

The navigation icons let you move to the next and previous file and the next and previous replacement.

The accept and reject icons let you accept or reject replacements on a change-bychange basis, file-by-file basis, or all at once.



Replacements that you have accepted are marked in the source display with green circles. Replacements that you have rejected are marked with red circles. Unresolved replacements are marked with question marks. You can click on a green circle to change it to red, and vice versa.



When you have finished resolving differences by either accepting them or rejecting them, click **Commit**. You are asked whether you are sure you want to make the replacement. The message shows how many replacements will be made.

If you are sure you want to make all the changes, click **Accept All** and then click **Commit Changes** in the All Changes Resolved message window.

If you decide not to make changes, you can click **Cancel** at any time. If you have accepted any replacements, you see a message that asks if you are sure you want to cancel without making replacements.

# **Entity Locator**

Not all entities fall into one of the tab categories shown in the *Entity Filter*. You can find and learn more about any entity by using the *Entity Locator*, which provides a filterable list of entities in the database. You can filter by name, by entity type, by where the entity is declared, within what container the entity is declared, or when the entity was last modified. You can also use architecture hierarchies to sort entities.

To open the *Entity Locator*, choose **Search > Find Entity** or **View > Entity Locator** from the main menu bar.

<ul> <li>Locator: All Entities</li> </ul>	(8 of 4865 entities	;)		9 B	×
Show: All Entities Sync 🕂					
Entity 🔶 🛓	Kind 🛓	Declared In 🛔	File 🛔	Date Modified	*
done ×	•			▼ 3/6/2012 4:56:29 PM ▼	
done	Local Object	do_flush	gzio.c	Friday, October 23, 2009 11:50:32 AM	_
finish_done	Enumerator	[unnamed]	deflate.c	Friday, October 23, 2009 11:50:32 AM	
gz_header_s::done	Public Object	gz_header_s	zlib.h	Friday, October 23, 2009 11:50:34 AM	Ξ
lOrigDone	Local Object	main	testzlib.c	Friday, October 23, 2009 11:50:18 AM	
lOrigDone	Local Object	main	testzlib.c	Friday, October 23, 2009 11:50:18 AM	-

As in other windows in *Understand*, when you right-click on an entity anywhere in the *Entity Locator*, a menu of commands available for the item appears.

If you check the **Sync** box, selecting entities in other *Understand* windows causes the Entity Locator to select that entity unless filters prevent the entity from being displayed.

Resizing Columns	Column widths can be sized to adjust how much of each column is visible. You can drag the column header divider between two columns to resize the column to the left. Or, double-click on the column header divider while the double-headed arrow is displayed and the field to the left of the divider will be expanded or shrunk to the maximum size needed to view all items in that column.
	You can right-click on a column header and choose <b>Freeze Column</b> to move that column to the left and prevent it from being repositioned. By default, the File column is frozen and is the left column.
Long versus Short Names	In the <i>Entity</i> , <i>Declared In</i> and <i>File</i> columns, you can right-click the column header or click the drop-down icon to specify the display format for entity names and filenames. For entities, you can choose the short or full name (which includes the name of the compilation unit). For filenames, you can choose the short, full, or relative path.

#### Column Headers

Column headers are tools in the *Entity Locator*. Left-click them to sort according to that column. Right-click a column or click the drop-down icon to see a menu that lets you control how entities are listed, sorted, and filtered.

Declared In	🛓 File 🕴 Date M	odified
	Entity Name as	9/29/20
anu reas	File Name as	13 3.58
anu.rea	Show Parameters as	13 3:58
gnu.rege	Contraction .	13 3:58
gnu.rege 💙	Sort Ascending	13 3:58
gnu.rege	Sort Descending	13 3:58
gnu.rege	Clear All Filters	13 3:58
gnu.rege	Filter Case Sensitivity	13 3:58
anu reas	Filter Pattern Syntax	13 3:58
gnu.rege	Freeze Column	13 3:58
gnu.regexp.r.	E RESYNTAX.JAVA ZIZZIZU	13 3:58

The entity list may be sorted by any column. Left-click on the column header to toggle between sorting in ascending order and descending order. The default sorting order is in ascending order of entity names.

```
Choosing Columns Click the + icon in the upper-right of the Entity Locator to see the Locator Column Chooser.
```

🔎 Locator Column Chooser - U
Columns
C Entity Kind
Declared In Declared In Kind
File Architecture
Date Modified
Metrics Columns
CountLineCode Cyclomatic
RatioCommentToCode
OK Cancel

The **Entity** column must always be displayed. You can enable or disable the other columns.

#### .....

Filtering the List

The field below each column heading lets you filter the entities shown by the *Entity Locator*. The filter can be entered manually or automatically based on what was right-clicked on.

For example, you may filter by the *Kind* column selecting a kind from the drop-down list. You can also right-click on any item listed in the *Kind* column and select **Filter By Selection** from the menu. This filters the list of entities to contain only entities of the kind you selected. The title bar shows how many entities match the filter.

Or, you can simply type a filter in any of the fields. To search for field values that do not contain a particular string, type ! (exclamation mark) and then the filter.

To clear a filter, just delete the text from the field in the column heading or right-click a column header and choose **Clear Filter** or **Clear All Filters**.

You can use the **Previous** and **Next** buttons to move through the history of filters you have used.



The following example shows Filter By Selection for an entity Kind:

To filter the Date Modified column, the left drop-down lets you select a comparison operator ( <, <=, =, >=, >), and the right drop-down lets you select a date from a calendar. You can modify the time by typing. You must select a comparison operator in addition to a date in order to filter the entities.

Similarly, the metrics columns allow you to filter with a comparison operator. For example, you can filter the entities to show only those with a Cyclomatic complexity greater than some value or a Comment-to-Code ratio less than some value.

Right-click a column or click the drop-down icon to see the context menu for that column. You can choose for the filter case sensitivity to be **Case Sensitive** or **Case Insensitive** (the default). You can also set the **Filter Pattern Syntax** to use fixed strings (the default), wildcards, or regular expressions.

- **Fixed string:** The string you type matches if that exact string is found anywhere in the column value.
- Wildcard: These are \* or ?, where \* matches any string of any length and ? matches a single character. For example, ??ext\_io matches any name having 8 letters and ending in *ext\_io*.
- **Regular expression:** A powerful and precise way to filter and manipulate text. You cannot use the **Case Sensitive** option if you are using regular expressions.

Using ! to search for field values that do not contain a particular string can be used with any Filter Pattern Syntax.

Symbol	Description	Example
٨	Match at the beginning of a line only.	^word
		Finds lines with word starting in the first column.
\$	Match at end of a line only.	word\$
		Finds lines that end with "word" (no white space follows word).
/<	Match at beginning of word only.	l <word< td=""></word<>
		Finds wordless and wordly but not fullword or awordinthemiddle.
1>	Match at end of word only.	l <word< td=""></word<>
		Finds keyword and sword but not wordless or awordinthemiddle.
	A period matches any single	w.rd
	character.	Finds lines containing word, ward, w3rd, forward, and so on, anywhere on the line.
*	Asterisk matches zero or more	word*
	occurrences of the previous character or expression.	Finds word, wor, work, and so on.
+	Match one or more occurrences of the previous character or expression.	<i>wor+d</i> Finds word, worrd, worrrd, and so on,
?	Match zero or one occurrences of the	wor?d
	previous character or expression.	Finds word and wod.

The following table lists some special characters used in regular expressions.

Symbol	Description	Example
[]	Match any one of the characters in brackets but no others.	[AZ] Finds any line that contains A or Z. [Kk][eE][Nn] Finds any variation of case when spelling "Ken" or "KEn" or "keN".
[^ ]	Match any character except those inside the brackets.	[^AZ ] Finds any line that does not contain the letters A or Z.
[-]	Match a range of characters.	[AZ] Finds any line containing letters A through Z on them but not lower case letters.
l	A vertical bar acts as an OR to combine two alternatives into a single expression.	word   let+er Finds word, leter, letter, lettter, and so on.
١	Make a regular-expression symbol a literal character.	\*/\$ Allows searching for *. This example finds all lines ending in */

A full explanation of regular expressions is beyond the scope of this manual. Unix users may refer to the manual page for *regex* using the command "*man -k regex*". For a comprehensive explanation of *regex* expressions we refer you to the book "*Mastering Regular Expressions*", published by O'Reilly and Associates (www.ora.com/catalog/regex).

# **Finding Windows**

If you have a number of windows open, you can use the options in the **Window** and **View** menus to organize or find particular windows.

You can close the current document window by choosing **Window > Close** *<current\_window>*. You can close all source files, graphical views, and other document windows by choosing **Window > Close All Document Windows**. If you have many windows open in the document area, you can right-click on the tab for the window you are using and choose **Close All, Close All But This, Close All Tabs to the Left**, or **Close All Tabs to the Right**.

If you choose **Window > Release Window**, the tabbed area changes to a separate window that can be resized and moved around your screen. You can select **Capture Window** from the drop-down icon in the upper-right corner of a released window to replace the window within the

main Understand window.



The Window menu also lets you use **Window > Split Vertically** or **Window > Split Horizontally** to split the document area. When the document area is split, new areas open in the half that has its box checked. You can drag tabs from one half of the document area to the other as needed. Choose **Window > Unsplit** to remove the split. You can use **Window > Tile** or **Window > Cascade** to arrange the open windows.

The **Window > Predefined Window Layouts** command lets you choose from several standard layouts for common tools. The layouts include the "Tight" layout, "Classic" layout, and "Multi-monitor" layout.

The Window > Windows command (Ctrl+Tab) opens a temporary list of currently open windows. When you double-click on an item in this list, the list goes away and focus is given to the item you chose. You can dismiss this area without choosing a window by pressing Esc.

	Show: All Windows	Show: All Windows		
	Dock Windows	Document Windows		
	Search Results for 'test' (4)	🐼 egrep.c		
	All Entities (463 of 463 entities)	🐼 try.c		
	P Architecture Browser	🔞 regexp.c		
	Scheduler Results	🐼 strpbrk.c		
	P Information Browser	Setting Started		
	Project Browser - fastgrep.udb			
	Type-Ahead Filter: Type to Filter	Alphabetic Sort		
	Show list. Or, you can check the <b>Type-Ahead F</b> characters in the name of the window you are lo <b>Sort</b> box sorts both the docked windows and the windows is filtered to match the string you type	n reduce the number of windows listed here by choosing a window type from the ist. Or, you can check the <b>Type-Ahead Filter</b> box and begin typing some ters in the name of the window you are looking for. Checking the <b>Alphabetic</b> box sorts both the docked windows and the document windows. The list of <i>is</i> is filtered to match the string you type after checking the box.		
Source Visiting History	You can move forward or backward through the code visiting locations using <b>Previous</b> and <b>Nex</b> This history is stored even between <i>Understance</i>	history of your source t icons in the toolbar.		
You can click the down-arrows to see the list of source locations in the histochemic choose <b>Clear History</b> from the drop-down Go Back list to clear the browsir				
	The source locations are stored as line numbers, not by entity name. If you want to save locations by entity rather than line number, see <i>Favorites</i> on page 142.			
View Menu Commands	If you have analyzed the project during this sest Log command to reopen the log.	sion, you can use the <b>View &gt; Analysis</b>		
	The View > Window Selector command opens windows. By default, this area lists only docume drop-down to change the type of window listed. underlined italics.	an area that lists currently open ent windows, but you can use the <b>Show</b> Any released windows are listed in		

Click a window name to make it active. By default, the Selector lists all windows, but you can choose to show only Editor windows or various other window types. The icons indicate the type of window, including whether the source file is unsaved.

▼ S	elector	9∂×
Shov	/: All Windows	<b>•</b> 1
P	Architecture Browser	
C.	deflate.c	
P	Functions (236 of 2080 entities)	
P	Getting Started	
C.	infback.c	
P	Information Browser	
C.	inftrees.c	
$\triangleright$	Locator: All Entities (386 of 2080 e	ntities)
P	Project Browser - zlib.udb	

When the Selector area is active, you can type a filter at the bottom of the area to quickly narrow the list. Press Backspace to erase the filter.

You can use the drop-down icon to change the order from alphabetic to most recently used or by file extension. You can also use the drop-down icon to change the filename format to show the short, relative, or absolute file paths.

Using the Selector is a convenient way to perform actions—such as Close—on multiple windows by selecting multiple windows from the list, right-clicking, and choosing **Close Selected Window(s)** or **Close Unselected Window(s)**.

If you have created bookmarks in your source code (page 181), you can use the **Bookmarks** command in the View menu to open the list of bookmarks.

## **Displaying Toolbars** You can hide or display categories of toolbar icons

You can hide or display categories of toolbar icons by rightclicking on the toolbar or menu bar and choosing a category. The toolbar is separated into the following categories: Project, File, Edit, Analyze, Editor History, Graphs, User Tools, Browse, Split Workspace, Scopes, and Search.

You can also hide and display toolbar sections by choosing **View > Toolbars** from the menus.



# Searching in a File

	The search techniques described in this section are used to search a single source file.			
Find Next and Previous	To search quickly within the current file, press Ctrl+F (or choose <b>Search &gt; Find</b> ). The status bar of the Source Editor changes to a search bar.			
	× public - 🤤 💭 Case Sensitive 🗌 Whole Words 🔲 Hide			
	You can type a string in the field. As you type, matches for that string are highlighted in the Source Editor. The drop-down list contains recent search strings.			
	Click <b>Previous</b> or <b>Next</b> to move from match to match. You can check the <b>Case</b> <b>Sensitive</b> and <b>Whole Words</b> boxes to modify how the search is performed.			
	If the string does not exist in the file, the search field turns red.			
	If you check the <b>Hide</b> box, then as soon as you click on the code, the incremental search bar is hidden. When you press Ctrl+F again, your last search is shown. Use Ctrl+Shift+F to find the previous occurrence.			
Find & Replace	If you want to use Search-and-Replace or regular expressions for searching within a single source code file, you can use the Find dialog. To open this dialog, choose the <b>Search &gt; Find &amp; Replace</b> menu item or press Ctrl+Alt+F.			
	Find			

Find		
Find:	•	
Replace:	▼	
Regular expression Match case Match whole words		
Replace	e All Replace Replace & Find Previous Next	

In the **Find** field, type the string you want to find.

You can check the **Regular expression**, **Match case**, and **Match whole words** boxes to modify how the search is performed. If you check the **Regular expression** box, you can use Unix-style pattern matching. For a list of some of the capabilities of regular expressions, see page 157.

If you want to replace the string you are finding, type that in the **Replace** field.

Click **Previous** or **Next** to search in either direction. Click **Replace All**, **Replace**, or **Replace & Find** if you want to replace the string that was found.

The Find dialog searches only individual files. To search multiple files, see *Find in Files* on page 150.

Contextual	The Contextu
Information Sidebar	but more pow

The Contextual Information Sidebar (CIS) is similar to the Scope List (see page 167), but more powerful. You can open the CIS by choosing **View > Contextual Information** from the menus. You can click the open or close the Contextual Information Sidebar.

→ Contextual Information Sidebar 🖉 🗗 🗙							
File Context deflate.c -				Scope Context deflateInit_			
Structure Browser	File Information			Scope Information	Context Browser		
Configuration_table	e			Function deflateInit_ 6	<b>V</b>		
EQUAL				Defined in: deflate.	c 💌		
Image: Contract of the section of	s		Ε	Return Type: int			
O dummy				Parameters		=	
UPDATE_HASH			F	Overloads		-	
M INSERT_STRING				▲ Calls			
CLEAR_HASH				b deflatelnit2_			
<ul> <li>deflatelnit_</li> </ul>				Called By			
eflatelnit2_		-		References		Ŧ	

The CIS shows the structure and information for the currently active Source Editor. The tabs in the CIS provide the following information:

- Structure Browser: This is an expanded scope list for the current file. It lists the names of structures in the file. In addition to functions, it lists includes, macros, classes, and more. The icon next to the name indicates the type of entity. If you point your mouse cursor at an item, the hover text shows the entity type and name. Press Ctrl+F to search within this tab.
- File Information: This tab provides an Information Browser for the current file.
- **Scope Information:** This tab provides an Information Browser for the current entity—that is, the one highlighted in the Structure Browser tab.
- **Context Browser:** This tab shows the current entity's location in the hierarchy on the left and the entities it contains on the right.

The switch icon (Ctrl+,) to the right of the File Information tab changes the current file in the Source Editor and the CIS to a file in the same directory with the same name but a different file extension (the "companion file" if such a file exists). For example, the switch icon can toggle from a \*.c or \*.cpp file to a \*.h file with the same name.

As always, right-clicking in any of these tabs provides links to more information about each entity.

# Chapter 6 Editing Your Source

This chapter covers Understand's source and text file editor.

This chapter contains the following sections:

Section	Page
Source Editor	166
Saving Source Code	170
Refactoring Tools	171
Other Editing Features	177
Annotations	184
Printing Source Views	190

# **Source Editor**

The **Source Editor** offers a full featured source code editor, with syntax coloring and right-click access to information most entities in your code.



The line numbers and "fold" markings to expand/collapse blocks of code can be turned on and off in the **Editor** category of the Understand Options dialog you can open with the **Tools > Options** command (see page 109). The display font and a number of other items can also be changed in the **Editor** category. You can also enable bookmarks, indent guide marking, and a right margin marker (page guide) in that category of the dialog.

You can zoom in or out to make the text larger or smaller by choosing one of the **View > Zoom** menu options.

The **Editor > Styles** category of the Understand Options dialog (see page 115) lets you change the colors used for different types of source code. The **Key Bindings** category (see page 104) shows a list (and lets you modify the list) of keystrokes you can use in the Editor.

#### Scope List

You can jump to a particular function, procedure, or other language-specific construct in the current source file by selecting from the scope drop-down list in the toolbar. The drop-down list shows all such constructs in the file the last time the project was analyzed.

You can click the + icon or choose View > Scope List to open the list in a Scope tab in the area where the Entity Filter is shown. This tab lists constructs in the current source file. This tab is useful for jumping around in large files.

•	-
_	float2rgbe
	rgbe2float
	rgbe_error
	rgbe_error_codes
	RGBE_ReadHeader
	RGBE_ReadPixels
	RGBE_ReadPixels_RLE
	RGBE_WriteBytes_RLE
	RGBE_WriteHeader
	RGBE_WritePixels
	RGBE_WritePixels_RLE

The numbers next to each name in the Scope tab are the line numbers where each entity is declared in the

file. Single-click on an item to view information about it in the Information Browser. Double-click on an item to jump to the location where that item is declared or created and to highlight all occurrences of that name in the current source file.

You can right-click on the Scope tab to choose a sort order from the context menu. The ascending and descending orders sort alphabetically or reverse alphabetically. The default is to sort by line number.



For more power than the scope list, use the *Contextual Information Sidebar* on page 164.

**Status Icons** Each file in a Source Editor has a status icon in its upper-left title bar. The

Each file in a Source Editor has a status icon in its upper-left title bar. The letter in the icon indicates the type of file. The icon color indicates whether the file has been modified but not yet analyzed. An asterisk by the filename means the file has unsaved changes.

- Yellow icon = analyzed project file (has not been modified)
  - Red icon = modified project file (needs to be analyzed)
- White icon = file not in the project

**Status Line** 

When a Source Editor is the active window, the status bar at the bottom of the *Understand* window shows the last time CodeCheck was run on this file, the line number and column number of the cursor position, the tab width setting for this file, whether the file is in read-write or read-only mode, and the source language.

CodeChecked: 4/24/2012 3:39:25 PM Line: 51 Column: 5 | Tab Width: 4 | RW | C

Go To Line Go To Line 42 OK Cancel

If you click the line number in the status bar (or choose **Search > Go to Line** from the menus), you can use the Go To Line dialog.

If you click the Tab Width, you can change the tab width for this file. See page 180. If you click the RW (read-write) indicator, it changes the mode to RO (read-only). If you click the language, you can choose which language this file is treated as using.

Selecting and<br/>Copying TextText can be selected (marked) then cut or copied into the Windows (or X11) clipboard.<br/>Selecting text works as standard for the operating system in use. On Windows,<br/>dragging while holding down the left mouse selects text. Alternately you can hold down<br/>the Shift key and move the cursor (via arrows or the mouse). Choose the Select All<br/>command in the Edit menu or the context menu to select the entire file.

If you hold down the Alt key (Ctrl key on X Windows), you can drag the mouse to select a rectangular area of source code—for example, to exclude tabs in the left margin from the copied text. You can also paste rectangular areas of code within the Source Editor.

```
15
       #define DO1(buf,i)
                           {adler += (buf)[i]; sum2 += adler;}
16
       #define DO2(buf,i)
                           DO1(buf,i); DO1(buf,i+1);
17
       #define DO4(buf,i)
                           DO2(buf,i); DO2(buf,i+2);
       #define DO8(buf,i)
18
                           DO4(buf,i); DO4(buf,i+4);
       #define DO16(buf)
                           DO8(buf,0); DO8(buf,8);
19
20
```

Once you select text, you can use the **Cut** and **Copy** commands in the **Edit** menu or the context menu. You may then paste the text into other applications as needed.

For entities with a class path and files, the **Copy** command copies the short name. The **Copy Full Name** command in the context menu copies the full class path or file path.

.....

You can switch a Source Editor to "Browse" mode by clicking the **Browse** button in the main toolbar or choosing **View > Browse Mode** from the menus. When you are in Browse mode, the icon is highlighted.



When you are in Browse Mode, entities in the code act as links. An underline is shown when your mouse cursor moves to a link. Clicking a link moves you to the declaration of that entity and updates the Information Browser to show details about that entity.

**Browse Mode** 

If the declaration of an entity you click on is not found, a message is shown in the status bar and your computer beeps.

When you are in Browse Mode, you can still edit the file and the keyboard and rightclick function the same as in regular mode. Only left-clicking the mouse is different.

You can temporarily enter Browse Mode by holding down the Ctrl key while using a Source Editor window. You can toggle Browse mode by pressing Ctrl+Alt+B.

See page 116 for settings to control the behavior of Browse Mode.

.....

The context menu in the Source Editor provides access to a number of exploration and editing features. Many of them let you find specific information about the entity you right-click on.

The following exploration features are typically included in the context menu (depending on where you click):

- View Information (see page 131)
- Graphical Views (see Chapter 10)
- Edit Source/Definition (see page 134)
- User Tools (see page 313)
- Explore (see page 139)
- Find in... (see page 150)
- Add Favorite (see page 142)
- Metrics Charts (see page 237)
- Metrics Browser (see page 233)

The following editing features are also typically included in the context menu:

• Undo / Redo

**Context Menu** 

- Cut / Copy / Paste (see page 168)
- Select All (see page 168)
- Jump to Matching Brace (see page 178)
- Select Block (see page 178)
- Hide/Show Inactive Lines (see page 178)
- Fold All (see page 178)
- Soft Wrap (see page 180)
- Comment Selection / Uncomment Selection (see page 179)
- Change Case (see page 179)
- Revert (see page 170)
- Add Bookmark (see page 181)

# **Saving Source Code**

If you have edited a source file, you can click  $\blacksquare$ , press Ctrl+S, or choose **File > Save** to save your changes.

You can choose **File > Save As** to save to a different file. If you save a project file to another filename, you will be asked whether you want to add the new file to the project.

If you have edited multiple source files, you can click kell or choose **File > Save All** to save changes to all modified files.

If you want to ignore changes you have made choose since the last save, right-click in a file and choose **Revert**.

You can close the current source file by choosing **Window > Close** <*current\_file>* from the menus. You can also middle-click on the tab above the source file area to close that tab (if your mouse has a middle button).

You can close all source files by choosing **Window > Close All Document Windows**. You can also right-click on the tab for the source file area and choose **Close**, **Close All**, **Close All But This**, or **Close All Tabs to the Right/Left**.

# **Refactoring Tools**

Refactoring tools allow you to make structural changes to your code. Ideally, refactoring does not change the behavior of the code.

The refactoring tools allow you to preview the changes using a code comparison window. Refactoring changes can have significant effects on code, and should be reviewed before committing to make certain that the changes are correct.



The following refactoring tools are provided:

- Rename: page 172 (various languages)
- Inline Function: page 173 (C/C++)
- Extract Function: page 174 (C/C++)
- Extract Temp: page 176 (C/C++)

If files have not been saved when you select a refactoring command, you are asked if you want to save the files and reanalyze the project. We recommend that you do this, so that the *Understand* database will contain current information about your project.

Modified File(s) Detected! - Understand - (Build 800)				
?	Would you like to save/analyze modified files before refactoring?. Don't show this message again. (Press Shift to see message again)			
	Save and <u>Analyze First</u> Continue Anyway Cancel			

If you perform a **Refactor** operation and then decide that you did not want that change, right-click and choose **Refactor > Undo** from the context menu. In some cases, the refactoring operation cannot be undone because of subsequent changes.

...........

**Renaming Entities** A command to rename entities is provided in order to support refactoring of your code in order to make your code more readable. It allows you to change the name of an entity throughout your project. The **Refactor > Rename** command is similar to **Replace in Files** (page 153). However, the difference is that the **Refactor > Rename** command determines which uses apply to that specific entity. This makes it a better way to rename such things as variables that are used locally, entities in applications with a variety of namespaces, and entities with names that may be a part of another entity name (for example, renaming a "src" variable without renaming "srcTimer").

To use the **Refactor > Rename** command, follow these steps:

- 1 Highlight the name of the entity (for example, a function or argument) to change.
- 2 Right-click and choose **Refactor > Rename** from the context menu.
- **3** Type a new name for this entity in the dialog and click **Preview Changes**. If you are absolutely sure you want to perform the renaming operation, click **Apply Changes**.

👰 Refactor: Rename - Understand - (Build 741)				
New Name:	destination			
	Preview Changes Apply Changes Cancel			

4 A dialog opens that lets you examine all the instances where this entity name is used and how it will be changed. Code Comparison on page 301 describes the icons and drop-down menus in this dialog.

🔎 C:\Progra	🙎 C:\Program Files\SciTools\sample\fastgrep\timer.c - Understand - (Build 739)				
Options, R	Options, Refresh [Current] [Refactored]				
🙌 Undo Cł	🙌 Undo Change 🗼 Redo Change h Prev 🕹 Next 1 of 7 Total Differences 🍦 Undo File 🥠 Redo File				
136	if (*fields.ans != 'c')	136		if (*fields.ans != 'c') 🔺	
137	complain("regcomp failu:	137		showWarning("regcomp fa:	
138	return;	138		return;	
139	- }	139	LF.	3	
140	if (*fields.ans == 'c') {	140	Þ	if (*fields.ans == 'c') {	
141	complain("unexpected regco	141		showWarning("unexpected r	
۰ III ا					
C:\Program Files\SciTools\sample\fastgrep\timer.c     Changed line 137     Changed line 141					

Inlining Functions Inlining functions is a common optimization technique that places the code of a function at the location where it is called instead of in a separate function. In compiled languages, this can often be performed through compiler optimization, but inlining the source code may be useful for various reasons, including code clarity. Note that inlining involves tradeoffs. If a function is called in many places, inlining the code results in a larger code size and less maintainable code.

To use the **Refactor > Inline Function** command, follow these steps:

- 1 Highlight the name of the function you want to inline.
- 2 Right-click and choose Refactor > Inline Function from the context menu.
- 3 Click **Preview Changes**. If you are absolutely sure you want to perform the inlining operation, click **Apply Changes**.



4 A dialog opens that lets you examine all the changes that will occur. Code Comparison on page 301 describes the icons and drop-down menus in this dialog.

C:\Program Files (x86)\SciTools\sample\zlib\deflate.c - Understand - (Build 800)					
Options, Refresh [Current]	[Refactored]				
🐳 Undo Change 🗼 Redo Change 🛛 🔶 Prev 🛛 🚽	🙌 Undo Change 🗼 Redo Change 👌 Prev 🛛 👆 Next 2 of 3 Total Differences 🆕 Undo File < 👘 Redo File				
509 /* default settings: return t	509 /* default settings: return (*				
510 return compressBound(sourceLe	510 return sourceLen + (sourceLer				
511 L	511				
512	512				
513					
C:\Program Files (x86)\SciTools\sample\zlib\compress.c					
Changed lines 75 - 80					
C:\Program Files (x86)\SciTools\sample\zlib\deflate.c					
Changed line 510					

**Extracting Functions** The opposite of function inlining is function extraction. You can extract some

**Ig Functions** The opposite of function inlining is function extraction. You can extract some code to a separate function so that it can be called in several places and maintained in one place.

To use the **Refactor > Extract Function** command, follow these steps:

- 1 Highlight the code that you want to extract as a function.
- 2 Right-click and choose **Refactor > Extract Function** from the context menu.

🔎 Extract Function - Understand - (Build 800)				
Function Name: my Return: void Parameters:	Function			
Entity	Name	Const	& _	Move Up
uLong total_out	total_out		<b>V</b>	Move Down
uLongf * destLen	destLen			,
Preview Changes Apply Changes Cancel				

- **3** Type a name for the function to be extracted and called from this code location.
- 4 Select the value or variable to be returned by the function.
- **5** For the parameters to be passed to the function, you can use the Move Up and Move Down buttons to change the sequence and check the boxes to identify parameters to be passed as const values or by reference.
- 6 Click **Preview Changes**. If you are absolutely sure you want to perform the operation, click **Apply Changes**.
- 7 A dialog opens that lets you examine all the changes that will occur. Code Comparison on page 301 describes the icons and drop-down menus in this dialog.



# Inline Temp Inline temp refactoring can be used with a local or temporary variable that is initialized and never set after that. The inlining replaces used of that variable with the expression to which it is initialized. In the following example, patlen could be inlined as altmin, so long as the value of patlen and altmin do not change between the initialization and any usage of patlen: int patlen = altmin;

```
for (k = str + patlen - 1; k < strend;) {
    ...
}</pre>
```

To use the **Refactor > Inline Temp** command, follow these steps:

- **1** Highlight the variable to inline.
- 2 Right-click and choose **Refactor > Inline Temp** from the context menu.
- 3 If you want the expression that replaces the variable to be surrounded by parentheses, check the box.
- 4 Click **Preview Changes**. If you are absolutely sure you want to perform the operation, click **Apply Changes**.



**5** A dialog opens that lets you examine all the changes Code Comparison on page 301 describes the icons and drop-down menus in this dialog.

Options, Refresh [Curren	nt]		[Refactored]	
🙌 Undo Change 🛛 🔶 Re	do Change 🛛 合 Prev	👆 Next	2 of 3 Total Differences	<mark> Undo File</mark> 🛛 🖓 Redo File
367		366		*
368 🖨 for	(k = str + patlen	- 1 367	for (k =	str + altmin - 1
369 🛱 /*		368	Ė /*	
370 *	for a large class	of 369	* for	a large class of T
				4
		11111111		
C:\Program Files\SciTools	s\sample\fastgrep\egrep.c			
Deleted line 354				
Changed line 368				
Changed line 411				

**Extract Temp** The opposite of inline temp is extract temp. If you have a complicated expression, you may want to assign a part of that expression to a local or temporary variable that can be reused within that function wherever the expression you select is used.

To use the **Refactor > Extract Temp** command, follow these steps:

- 1 Highlight the expression you would like to extract to a local or temporary variable.
- 2 Right-click and choose **Refactor > Extract Temp** from the context menu.
- **3** Type a name for the extracted variable in the dialog and click **Preview Changes**. If you are absolutely sure you want to perform the operation, click **Apply Changes**.



**4** A dialog opens that lets you examine all the changes Code Comparison on page 301 describes the icons and drop-down menus in this dialog.

C:\Program Files\SciTools\sample\fastgrep\egrep.c - Understand - (Build 800)									
Optic	ons, F Undo C	Refresh Change	] [Current]	🔶 Prev	÷	Next	[ 1 c	Refactored] of 2 Total Differences 🏼 🆓 Undo File	Redo File
309	Э	-				318		sizeIncrement = patsta	at.st_s: ^
310	C	mall	Loc((unsigned)	patstat.st	_	311		malloc((unsigned) size	Increme
311	1	of	memory to read	d pattern f	<b>::</b> ]	312		of memory to read pat	tern fi
312	2	.st	size !=		4	313		.st_size !=	
313	3	at,	<pre>sizeof(char),</pre>	patstat.st	_	314		at, sizeof(char), size	Increme
21/	1	- n 1	coding notton	e filolly		215		or reading pattorn file	~ <b>II</b> \.
•									•
▲ C:\	\Progra	am Files\	SciTools\sample\fastgr	ep\egrep.c					
Changed line 310									
Changed line 313									
								ОК	Cancel

# **Other Editing Features**

The Source Editor also provides several other options for displaying and editing files:

- Previewer on page 177
- Bracket Matching on page 178
- Folding and Hiding on page 178
- Splitting the Editor Window on page 179
- Commenting and Uncommenting on page 179
- Changing Case on page 179
- Indentation on page 180
- Line Wrapping on page 180
- Insert and Overtype Modes on page 180
- Sorting Lines Alphabetically on page 180
- Keyboard Commands on page 180
- Recording, Playing, and Saving Macros on page 180
- Creating and Opening Files on page 181
- Bookmarking on page 181
- Managing Source Editor Tabs on page 183
- Changing the Source Code Font Size on page 183

### Previewer Th

The Previewer window is similar to a Source Editor window. To open the Previewer window, choose **View > Previewer**. The differences between the Previewer and the Source Editor are as follows:

- You cannot edit the code in the Previewer window.
- **Sync checkbox**; If this box is checked, a single-click on an entity in another view displays the location where that entity is defined in the Previewer.
- **Prefer checkbox**; If this box is checked, a double-click on an entity in another view displays the location where that entity is defined in the Previewer. (Double-clicking on an entity in the Previewer always opens that entity's definition in the Source Editor.)



Bracket Matching	A handy feature of the <i>Understand</i> editor is syntax bracket matching. Use this feature to find the matching ending character for a brace, parenthesis or bracket. Symbols matched are (), { }, and []. Matching isn't done inside comments.					
	Pressing Ctrl+j (or right-click and <b>Jump to Matching Brace</b> ) jumps the editor to the matching end or beginning brace. Ctrl+j isn't active unless your editing cursor is by a symbol that it can match. Another Ctrl+j takes you back where you started. You can also choose <b>Search &gt; Go to Matching Brace</b> from the menus.					
	Pressing Ctrl+Shift+J (or right-click and <b>Select Block</b> ) selects all the text from the bracket to its matching bracket. Brackets without a match are highlighted in red when you move your cursor to them. Brackets with a match are highlighted in green.					
	When your cursor is on a preprocessor directive that has a match (for example, #ifdef and #endif), you can use Ctrl+j (or right-click and <b>Jump to Matching Directive</b> ) to move your editing cursor to the match.					
Folding and Hiding	The - and + markings next to the line numbers allow you to "fold" the code to hide blocks such as functions, if statements, and other statements that have a beginning and end.					
	7 = #ifndef std_include 8 # #ifndefnested 15 = #endif					
	If you right-click on the code, you can choose <b>Fold All</b> to close all the open blocks. You can also fold and unfold source code by choosing <b>View &gt; Fold All</b> from the menus.					
	You can add explicit fold markers to code in languages where // is treated as the					

You can add explicit fold markers to code in languages where // is treated as the beginning of a comment. For example:

```
//{{
    /* code to hide when folded */
    //}}
```

You can also choose **Hide Inactive Lines** to hide preprocessor lines that are not active because a preprocessor macro is not defined. Choose **Show Inactive Lines** to view all lines again. You can also toggle this setting by choosing **View > Hide Inactive Lines** from the menus.

 Splitting the Editor
 You can click the Split icon (circled below) to divide the source editor into two or more separately scrollable panes. Click one of the Join icons to merge two panes.

 Image: Comparison of the Compari



Indentation	You can click <b>Tab Width</b> in the status bar at the bottom of the window to open a dialog that lets you set the number of columns for each tab stop in this file. This setting is saved separately for each file, and overrides the setting in the Editor category in the Options dialog (see page 109).						
	CodeChecked: 4/24/2012 3:39:25 PM Line: 51 Column: 5 Tab Width: 4 RW C						
	You can make the indentation of selected code match standard usage by selecting the code, right-clicking, and choosing <b>Reindent Selection</b> . Indentation preferences are controlled by the <b>Editor &gt; Advanced</b> category in the Options dialog (see page 111).						
Line Wrapping	Normally, lines are cut off on the right if your Source Editor window is not wide enough to display the full line length. You can make the Source Editor wrap long lines to display all the code. To do this, right-click in the Source Editor and choose <b>Soft Wrap</b> . You can also change the wrapping mode by choosing <b>View &gt; Soft Wrap</b> from the menus. The wrapping is for display only; no actual line breaks are added to your source file.						
	See <i>Editor &gt; Advanced Category</i> on page 111 to change the wrap mode for source code printing.						
Insert and Overtype Modes	Normally, text to the right of your typing cursor is shifted as you type. This is called Insert mode. To switch between Insert mode and Overtype mode, in which text to the right of the cursor is replaced character-by-character as you type, press the <b>Insert</b> key or choose E <b>dit &gt; Toggle Overtype</b> from the menus.						
Sorting Lines Alphabetically	To sort a group of lines into alphabetical order, select the lines, right-click and choose <b>Sort Selection</b> .						
Keyboard Commands	To see a list of keystrokes that work in the Source Editor, choose <b>Tools &gt; Options</b> and go to the <b>Key Bindings</b> category. For example, Ctrl+Alt+K cuts the text from the cursor position to the end of the line. And, Ctrl+T transposes the line at the cursor position with the line above it.						
	Another way to see a list of key bindings is to choose <b>Help &gt; Key Bindings</b> . Search for the line that says "Editor" (around line 110) to get to the beginning of the keystrokes for the Source Editor windows.						
Recording, Playing, and Saving Macros	You can record and replay a set of editing changes that you want to be able to repeat. These are called macros. To record a macro, follow these steps:						
	<ol> <li>Choose Tools &gt; Editor Macros &gt; Record Macro from the menus or press Ctrl+Alt+M.</li> </ol>						
	2 Perform the steps you want to be able to repeat in the Source Editor.						
	3 Choose Tools > Editor Macros > Stop Recording or press Ctrl+Alt+M. (Note that if your cursor is not in the Source Editor at the end of the macro, you will not be able to stop the recording until you move back to the Source Editor.)						
To replay the most recently recorded macro, move your cursor to the desired start location and choose Tools > Editor Macros > Replay Macro or press Ctrl+M.

You can save the most recently recorded macro by choosing Tools > Editor Macros > Save Macro or pressing Ctrl+Shift+M. You will be asked to type a name for the macro. You can also move to the Shortcut field and press the key combination you want to use to trigger this macro.

🙆 Save Macro	o - Understand	I 💡	×		
Macro Name:	Replace Test2				
Shortcut: Ctrl+Alt+T					
SEARCHANC	HOR				
SEARCHNEX	T "test"				
SEARCHANCHOR					
SEARCHNEXT "test"					
REPLACESEL "t"					
REPLACESEL "e"					
REPLACESEL "s"					
REPLACESEL "t"					
REPLACESEL "2"					
	Save	Cance			

You can rename and delete saved macros in the Understand Options dialog by choosing Tools > Editor Macros > Configure Macros. See page 114 for details.

Creating and Opening Files	You can use the Source Editor to create an untitled blank file by choosing <b>File &gt; New &gt;</b> <b>File</b> from the menus. You can open files, whether they are in your project or not, by choosing <b>File &gt; Open &gt; File</b> .
	When you right-click on a filename, the context menu provides options to <b>Edit File</b> and to <b>Edit Companion File</b> . For example, the companion file of encrypt.c is encrypt.h.
Bookmarking	You can create "bookmarks" in your code by right-clicking on a line and choosing Add Bookmark from the context menu. Or choose Edit > Bookmarks > Toggle Bookmark from the menus. Lines with a bookmark have a red arrow next to them.
	In a file with multiple bookmarks, you can right-click and choose <b>Previous Bookmark</b> or <b>Next Bookmark</b> to guickly move between places in a file. These commands are also

available under Edit > Bookmarks in the menus.

You can open a Bookmarks area to view a list of all your bookmarks in all your files by choosing **View > Bookmarks** from the menus.

✓ Bookmarks - C:\AppData\Roaming\SciTools\sample\fastgrep\fastgrep.udb				
View by: 📔 🗾 📝 📟 🗦 🚍 🔀				
4 📁 No Category				
testzlib.c: 152 (Function: main)				
function Deflate_Init				
4 📁 Fixes				
> zip.c: 293				
🔺 📁 To Do				
gzjoin.c: 106 (Static Function: bopen)				
✓ zlib-thin.adb: 30 (Package: Thin)				
function Avail_Out (Strm : in Z_Stream) return UInt is				

If you point to bookmarked code in the Bookmarks area, the 5 lines of code surrounding the bookmarked line are shown in the hover text.

Double-click on a bookmark to move to that location in the Source Editor. If you create a bookmark *inside* an entity, the Bookmarks area shows the name and type of entity that contains the bookmark. For example, the function name is shown if you create the bookmark on the first line of code inside a function.

Bookmarks and Favorites (page 142) are stored as part of the project. If you want to mark code locations on a cross-project basis, see *Annotations* on page 184.

The toolbar for this area lets you manage your bookmarks in the following ways:

View by: D Mou can use the View by icons to switch between a file-based and a category-based view. The file-base view lets you expand filenames to see the bookmarks in that file. The category-based view lets you assign bookmarks to categories you create.

Select a bookmark and click this icon to change the category the bookmark is in. To create a new category, type the name and click **OK**. To use an existing category, select it from the list.

Select a bookmark and click this icon to delete that bookmark.

Select a bookmark and click this icon to mark it as a temporary bookmark to be deleted 24 hours after marking it as temporary.

Select a file in the file-based view and click this icon to delete all the bookmarks in this file. You can also select a bookmark and click this icon to delete all the bookmarks in the file that contains the selected bookmark.

Select a category in the category-based view and click this icon to delete all the bookmarks in the category. The category itself is not deleted.

Click this icon to delete all your bookmarks.

The drop-down icon in the upper-right corner of the Bookmarks area provides the following commands:

- **Copy to Clipboard on Double Click:** By default, double-clicking on a line in the Bookmarks list jumps to that location in the code. If you enable this option, double-clicking both jumps to the location in the code and copies that line of code to your clipboard.
- **Show Original Indentation:** Enable this option to display the code line with indentation matching the source code indentation.

Managing SourceWhen you right-click on the tab at the top of a Source Editor, some of the commands<br/>allow you to control the behavior of the tab.



If you choose **Show Tab Title as**, you can shorten or lengthen the filename in Source Editor tabs. Likewise, if you choose **Show Window Title as**, you can shorten or lengthen the filename in the *Understand* title bar and any separate Source Editor windows. The **Copy Filename** command lets you copy the long, relative, or short filename to the clipboard.

If you choose **Release Window**, the tabbed area changes to a separate window that can be moved around your screen. Click do to change a tab to a window within the *Understand* window.

Changing the Source<br/>Code Font SizeYou can change the default display font and font size in the Editor category of the<br/>Options dialog that you open with the Tools > Options command (see page 109).In addition, you can change the display size of the font for an individual source code<br/>window by choosing options from the View > Zoom submenu. View > Zoom > Zoom<br/>In makes the font size larger. View > Zoom > Zoom Out makes the font size smaller.<br/>View > Zoom > Reset Zoom changes the font size back to the default.

#### Annotations

Annotations let you add comments or notes about entities without changing the source code directly. You can view the annotations inline, following the definition of the entity to which they are attached. They can also be seen in hover text wherever the annotated entity is used, including in graphs and in the Information Browser.

295	FILE *pf;
296	struct stat patstat;
	<pre>#issue:CQ0034567 Enhance to provide more statistics</pre>
	Me - Tuesday, August 21, 2012 11:40:06 AM
297	<pre>static char *pat;</pre>
298	

Each annotation can be "tagged" with a key value pair. Such tagging is useful for organizing your notes using keywords, author names, or any other identifier you want to use.

Adding an Annotation To add an annotation, follow these steps:

- 1 Highlight an entity, such as a variable or function name, anywhere in *Understand*. For example, you can select an entity in source code, in the Information Browser, or in the Entity Filter.
- 2 Right-click on the entity and choose **Annotate** and the entity or line number you want to annotate from the context menu. (Line numbers cannot be annotated in a file that has not been saved. Annotations to a line number are updated if possible when the line number changes.)

3		<pre>char * strpbrk(pat1, pat;</pre>	2)				
4 5		register char *pat1;		View Information			
6		char *pat2;		Graphical Views	١.	ear	ch for */
7	P	{		Interactive Reports	ŀ.		
8				Edit Definition	×	1	
9		register char *cpl,		Add to Favorites	F		
11		cp1 = pat2;		Add Selection to Favorites	×		
12	Ц			Add Location to Favorites	×		
13	님	while (*cp1 != '\0')		Annotate	F		Annotate pat2
15		$\operatorname{return}(\operatorname{cp2}):$		Remove Annotations			Annotate strpbrk
16		cp1++;		User Tools	×		Annotate strpbrk.c
17		}		Explore	×		Annotate Line 3

3 If this is the first time you are adding an annotation, you will see the Annotations Setup dialog. Type your name in the **Author name** field. The annotations are stored in a \*.ann file within the project. The **Default Annotations File** is stored in the project directory. The filename includes your name and the name of the project. (See page 186 to learn about managing annotation files.)

🚱 Annotations Setup	? ×
To annotate, you must have an author name, and pick a annotations file where new annotations will be saved.	a default
Author name: Me	
Default Annotations File: /fastgrep/Me-fastgrep.ann	Browse
ОК	Cancel

4 In the Add/Edit Annotations dialog, type your annotation comment in the right box. If you want to tag your comment so that it will be easy to search for, begin the text with a #key or #key:value tag. For example, you can use #reminder:CodeReview to flag items that should be reviewed. Or, you could use #errorchecks to flag items that need to have their status tested. The author name you entered is automatically associated with your annotations, so you don't need to include your name in a #key:value pair. You can type multiple keys in a single annotation, and keys can occur anywhere within the annotation text. If you want to use a # sign in the annotation text without having it treated as a key, type ##.

Add/Edit Annotations for pat	tstat ? X
🕂 🗕 Tuesday, August 21, 2012 1	Add or select an annotation to edit. You can tag an annotation using #key:value or #key. Tags may not contain whitespace or quotes. Use ## for the # symbol.
	#issue:CQ0034567 Enhance to provide more statistics
	Save Cancel

- 5 If you want to create multiple annotations for the same item, click the "+" icon. The current date and time are added to the left box. You can select annotations using this timestamp when you want to edit annotations.
- 6 Click **Save**. Your annotation appears in the source code where the selected entity is defined. (See page 186 to control how annotations are displayed.)

You can also add an annotation by clicking on an entity and choosing **Annotations** > **Annotate** > **Annotate** <**entity**> from the menus. Options are shown to annotate the current file, the current entity, and any entities that contain the current entity (such as a function that contains the selected variable).

Editing an AnnotationTo edit an existing annotation, follow these steps:Annotation1Highlight an entity that has an annotation anywhere in Understand.2Right-click on the entity and choose Annotate and the entity you want to annotation	Э
<ul> <li>Annotation</li> <li>1 Highlight an entity that has an annotation anywhere in <i>Understand</i>.</li> <li>2 Right-click on the entity and choose Annotate and the entity you want to annotate from the context means.</li> </ul>	9
2 Right-click on the entity and choose <b>Annotate</b> and the entity you want to annotat	e
from the context menu.	
3 In the left box, select the timestamp for the annotation you want to edit.	
4 Modify the annotation text in the right box.	
5 Click <b>Save</b> to store your changes.	
If you edit an annotation that was originally created by someone else, that other pers remains the author of the annotation. The timestamp for the annotation is updated to the last time it was edited.	วท
<b>Deleting an</b> To delete an existing annotation, follow these steps:	
Annotation 1 Highlight an entity that has an annotation anywhere in <i>Understand</i> .	
2 Right-click on the entity and choose Annotate and the entity whose annotation y want to delete from the context menu.	ou
3 In the left box, select the timestamp for the annotation you want to edit.	
4 Click the "-" (minus) icon above the list of timestamps.	
5 Click <b>Save</b> to finish deleting the annotation.	
ManagingYou can choose Annotations > Annotation Options from the menus, and then set tAnnotation Files andfollowing options:	ne
• Your name or username to identify the original author of your annotations.	
<ul> <li>The annotation files to look in for this project, and which file is the default for annotations you add. For example, you can have Understand display annotations from separate files for everyone working on this project.</li> </ul>	i
<ul> <li>The foreground and background colors to use when displaying annotations from each of the files.</li> </ul>	
<ul> <li>How to display annotations: inline, as hover text, and with an indicator.</li> </ul>	
See Annotations on page 59 for details about setting these options.	
If other developers are also annotating code using <i>Understand</i> , choose <b>Annotations Refresh Annotations</b> from the menus when you want to get the latest annotations the have added.	, > ey

SearchingYou can search for annotations based on the key:value pairs, the author, and the<br/>timestamp. To search annotations, follow these steps:

- 1 Choose Annotations > Search Annotations from the menus.
- 2 Specify any of the following search parameters you want to use.

Annotations	? ×
Filters Search Annotations	Manage Orphaned Annotations
Show appotations with date later t	than: 12/1/2012 12:00:00 AM
Show annotations with date earlie	er than: 12/1/2012 12:00:00 AM
<ul> <li>Include Orphans</li> <li>Filter values should be a comma seperative refers to tags without a value such as #</li> </ul>	Search Only Filtered Annotations ted list of allowed values. "no value" #tag.
All	Add Filter Add Filter Group
author	▼ Chris
issue  has values	▼ 42
	Close Search

- **Date range:** Check one or both date range boxes if you want to find annotations edited after a certain date and time and/or annotations last edited before a certain date and time.
- **Include Orphans:** Check this box if you want the search to also find annotations that are linked to entities that have been deleted. See page 189 for more about orphaned annotations.
- Search Only Filtered Annotations: Check this box if you want to limit the search to annotations that match the current filters. See page 188 for more about filtering annotations.
- Filter values: You can set up one or more filters for the search based on the author and any #key:value pairs in the annotations. The "has values" and "doesn't have value" options let you type a value to match or exclude for a #key:value pair. Exact matches for the author name and key values must be used; partial matches and wildcards are not supported. The "any value" option matches any annotation that has that #key, no matter what the value. The "no value" option matches annotations that have that #key, but no #key:value pair. Use Add Filter to create another filter, and choose All or Any to determine how matching is performed.

You can even use **Add Filter Group** to create nested levels of filters that have different settings for **All** and **Any**.

- 3 Click **Search**. The results are shown in the Annotation Search Results area in the main *Understand* window.
- 4 Expand the search results, and double-click on an item to go to the location where that annotation appears in the code. (That is, the location where the entity associated with the annotation is defined.)

Annotations are stored in \*.ann files, which use the SQLite database format. In addition to viewing annotations in *Understand*, you can use other applications that support SQLite to modify and search annotation files.

If other developers are also annotating code using *Understand*, choose **Annotations** > **Refresh Annotations** from the menus when you want to get the latest annotations they have added.

**Filtering Annotations** You can filter annotations based on the key:value pairs, the author, and the timestamp. To filter annotations, follow these steps:

- 1 Choose **Annotations > Filter Annotations** from the menus.
- 2 Specify any of the following filters you want to use.

Annotations		? ×				
Filters	Search Annotations	Manage Orphaned Annotations				
Show ann	otations with date later th	an: 1/1/2000 12:00:00 AM 👻				
Show ann	otations with date earlier	than: 9/1/2012 12:00:00 AM 👻				
Filter values should be a comma seperated list of allowed values. "no value" refers to tags without a value such as #tag.						
X issue	e   Ino value or  doesn't have value	values  Chris				
	ОК	Cancel Apply				

- **Date range:** Check one or both date range boxes if you want to find annotations edited after a certain date and time and/or annotations last edited before a certain date and time.

- Filter values: You can set up one or more filters for the search based on the author and any #key:value pairs in the annotations. See page 187 for details on using these fields.
- 3 Click **OK** or **Apply**. The filters you specify are applied to the annotations shown throughout Understand.

**Managing Orphaned** If you create an annotation, and later delete the entity with which it was associated, that annotation becomes an "orphan" when you re-analyze the project. Orphan annotations aren't shown in the code anywhere. You can manage orphan annotations by choosing whether to delete or re-attach them. To manage orphan annotations, follow these steps:

1 Choose Annotations > Manage Orphaned Annotations from the menus.

Annotations				
Filters	Search Annotations	Manage (	Orphaned Annotations	
Select an orpha entity on the rigl annotation, only	n on the left and an entity ht, all the annotations for t that annotation will be at	on the right that entity w ached.	. If you attach an orphan ill be attached. If you sel	entity to an ect a single
Orphans		Show:	Functions	▼ 1
⊳ k		Filter:		
4 Led_conne Christ	ectHandler	Led_co	nnectDevice	- E
Chris:#ennance:configure device		Led_co	nnectHandler	
		Led_dis	sconnectHandler	
		Led_led	IState_fetch	
		Led_led	IState_store	~
	Delete Orphan		Attach Orphan To	Entity
		0	K Cancel	Apply

- 2 Expand an orphan in the list on the left to see the annotation text.
- 3 If you want to delete the selected annotation, click **Delete Orphan**.
- 4 If you want to attach the selected annotation to a different entity, select an entity from the list on the right. (You can shorten the list by selecting a type of entity from the **Show** drop-down.)
- 5 Click Attach Orphan To Entity to connect the selected orphan to the selected entity. The annotation will be shown in the code where the new entity is defined.
- 6 Click OK.

Annotations

## **Printing Source Views**

The menu option **File > Print** opens the standard print dialog for your operating system so you can print the currently viewed source file. The output shows 66 lines per page.

By default, files are printed in the font and color shown on the screen when you choose the **File > Print** menu option. You can customize code printing in the Options dialog. To open this dialog, choose **Tools > Options**. Expand the **Editor** category, and select the **Advanced** category. Options to control how code is printed are in the Print area. See *Editor > Advanced Category* on page 111 for details about these fields.

Print
Font Size: 10 🚔
Color Mode: Normal
Wrap Mode: Wrap Word
Print absolute file name

To change the print output without changing the online display, choose the **File > Page Setup** from the menus. This dialog offers printing options similar to the following; the options may differ depending on your operating system:

Page Setup	
	Class of the spectrum     1       Status of the spectrum     1
Paper	
Size: Let	tter (8 1/2 x 11 in)
<u>S</u> ource: Sh	eet 💌
Orientation	Margins (inches)
Portrait	Left: 0.116 <u>Right:</u> 0.116
C Landscape	<u>T</u> op: 0.116 <u>B</u> ottom: 0.116

## Chapter 7 Architecting Your Codebase

This chapter explains the architecture features provided by *Understand* and explains how you can use them to analyze your code.

This chapter contains the following sections:

Section	Page
About Architectures	192
Using the Architecture Browser	193
Viewing Architecture Dependency Graphs	196
Viewing Architecture Metrics	200
Managing Architectures	201
Creating an Architecture	202
Editing an Architecture	204
Using XML to Manage Architectures	206

#### **About Architectures**

An architecture is an abstract hierarchy layered onto a body of source code. For example, a staff architecture could have nodes for each engineer working on a particular project. The nodes would contain a list of source code files belonging to or to be modified by that engineer. Dependencies and interactions could then be derived from that architecture.

Architectures allow you to name regions of a software project or ways of looking at software hierarchically. An architecture creates a hierarchy of source code units (entities). You can use the provided architectures or create your own.

Architectures need not reference every source entity in the database; that is, they can define a subset of the entities. Also, architectures can contain a particular entity more than once. (Technically, that is, the architecture's flattened expansion need not maintain the set property.)

You can combine architectures successively to create novel filters for your entities.

From a more technical perspective, simple set algebra is used to combine and transform architecture hierarchies. The result of the filter is a list of entities. This result list can be viewed as a flat list or in terms of another architecture. The filter definition can be saved as a dynamic architecture. A dynamic filter architecture is updated as the contents of the database change and it can be used to reconstitute the filter at a later date.

#### Using the Architecture Browser

To open the Architecture Browser, choose **Project > Architectures > Browse Architectures** from the main menu bar.



You see an expandable list of the architectures currently defined for your project.

This Architectures area is similar to the Filters area. When you click on an item, information about it is automatically shown in the Information Browser (as long as the "Sync" box is checked in the Information Browser).

Exploring Architectures

To explore the existing architectures, click the "+" signs to expand the hierarchy. Entities, such as files, functions, and variables are shown in the hierarchies.

Understand provides some "auto-architectures" that are built in:

- **Directory Structure:** Lists the project files in their normal file hierarchy—showing directories and their subdirectories.
- Calendar: Lists files in the project according to their last change date. A hierarchy of dates is shown that progresses from This Year, This Quarter, This Month, and This Week to Yesterday and Today.
- Language: Lists files first by their source code language and then by their location in the directory structure. (This architecture exists only if your project contains multiple languages.)
- Visual Studio Projects: This architecture is provided only if the project is configured to contain a Visual Studio project. (Existing projects must be reanalyzed for this architecture to be created.)

The auto-architectures are updated only when the project is analyzed. So, if your source code is actively being modified and you have not analyzed it recently, architectures—especially the Calendar architecture—could be out-of-date.

As always, you can right-click on any item in the Architecture Browser to get a list of information you can view about that item.

#### **Right-click on file in Architecture**

#### Right-click on Architecture node

▼ Architecture Browser		≰ ▼ Architecture Browser			
▲ Calendar		⊿ Ca	alendar	r	
▲ Earlier		4	Earlie		
adler32.c (Fil	e)		a		View Information
Assembl	View Information		Α		Add to Favorites
blast.c (I blast.b (I	Graphical Views		b		Graphical Views
Checksu	CodeCheck •		c		CodeCheck •
CircularE	View Dependencies		С		View Dependencies
CodecBa	Interactive Reports		C		Interactive Reports
crc32.c	Edit File		c	122	Metrics Summary
crc32.h	Edit Companion File(s)		с	125	Metrics Export
crypt.h (			c		include Expert
deflate.c	Rename File		d		XML Export
deflate.h	Add to Favorites		d		Annotate
Deflater.	Analyze adler32 c		D		liser Tools
DotZLID.			U 0		
example	Remove from Project		с а		Find "put4" in Calendar/Earlier
fitblk c (F	Annotate •		fi		Metrics Charts
gun.c (F	Remove Annotations		g		Browse Metrics
gvmat32	User Tools		g	D	Duplicate Architecture
gzappen gzip.c.(F Explore			g	2	Manage Architectures
GZipStre	Find "put4" in C:\Program Files (x86)\SciTools\		Ģ	ZinS	tream cs (File)
gzjoin.c					
gzlog.c (					
gzlog.h (	Browse Metrics				

infback.c (File)

Notice that the context menu for an architecture node (such as a filesystem directory or "This Quarter" contains some extra items not available in other context menus:

- **Graphical Views > Graph Architecture:** Creates a graph of the architecture hierarchy from this point down. You are asked whether you want to include entities in the graph or just the architecture nodes. See page 196.
- Graphical Views > Dependency Graphs: Shows the dependencies between architecture nodes. See page 196.
- **Metrics Summary:** Provides metrics for the entities within the selected node. The metrics are based on entities in the current node, but not those in sub-nodes lower in the hierarchy. See page 200.

- **Metrics Export:** Creates a CSV output of the metrics from the Metrics Summary. See page 200.
- **XML Export:** Creates an XML export listing the architecture nodes and entities from the selected point down in the hierarchy. See page 206.
- Edit Architecture: Opens the Architecture Builder for the selected architecture if it is one you created. You cannot edit the auto-architectures provided with *Understand*. See page 204.
- **Rename Architecture:** Opens a Rename Architecture window that lets you rename the selected architecture or node if it is one you created. You cannot rename the auto-architectures provided with *Understand*. See page 202.
- **Duplicate Architecture:** Opens a Duplicate Architecture window that lets you type a name for a duplicate copy of the selected architecture. See page 202.
- Manage Architectures: Opens the Architect Manager window. See page 201.

#### **Viewing Architecture Dependency Graphs**

You can generate graphs that show the hierarchy of an architecture. You can save these graphs as PNG, JPEG, SVG, Visio XML, and DOT files.

*Note:* Dependency graphs are also available for classes and packages.

To create a graph, follow these steps:

- 1 Select the highest-level architecture node you want to graph. You can graph the entire hierarchy or just a sub-hierarchy.
- 2 Right-click on the node and choose Graphical Views from the context menu. Depending on the node you select, the submenu allows you to choose Graph Architecture, Depends On, Depended On By, Butterfly-Dependency Graph, and Internal Dependencies. When you have selected an architecture node, the same list of graphical views is available by choosing Graphs > Graphs for <current\_entity> from the menus.

To open the Internal Dependencies graph for an entire architecture, choose from the **Graphs > Dependency Graphs** menu.

Architecture dependency graphs have the same toolbar as other types of graphical views. See page 250 for details about using the icons in the graphical view toolbar.



To save a graph as a JPG, PNG, or SVG file, see page 276. To save a graph to a Visio file, see page 276.

Dependency and relationship graphs provide an additional Graph Customizer toolbar that you can use to modify the graph display. This toolbar lets you control expansion, highlighting, and arrows on a per-node basis. It also lets you undo and redo your changes, and save and load graph customizations. For example, this is the default Depended On By graph for the C|C++ node in the multi-language zlib sample project.

Nodes that are drawn as 3D boxes (like those in this figure) can be expanded to shown the nodes they contain by double-clicking on them. You can keep expanding nodes until you get to the file level.



You can hover your mouse cursor over a line that connects two boxes to see which items are connected by the relationship and to highlight the line. (See page 118 if highlighting is disabled.)

Click the **View Dependencies** button or right-click on a dependency graph and choose **View Dependencies** to open the Dependency Browser (page 140) for the selected node or relationship. If you check the **Sync** box in the Dependency Browser, it shows details about any relationship you select in the graph, and the two nodes connected by the relationship are highlighted in the Dependency Browser. In addition, **Show** and **Group By** settings from the dependency graph are synced with the Dependency Browser and the **Dependency Kind** is changed to Depends On.

	You can right-click on the gray background of a dependency graph (outside any cold boxes) to control whether long, short, or relative names are displayed for architectu node names and filenames. In addition, you can enable or disable the reference co numbers that show how many times a particular dependency occurs. See Controlli Cluster Graph Layout on page 272 for details about the context menu options withi architecture dependency graphs.					
	The context menu when you right-click on a node in a dependency graph offers commands similar to those you see when right-clicking on an entity or architecture node name elsewhere in <i>Understand</i> .					
	The context menu when you right-click on an eprovides a list of the references that constitute visit the source code for this relationship. You described for the <b>Tools &gt; Options</b> dialog on p	ontext menu when you right-click on an edge (arrow) in a dependency graph les a list of the references that constitute the edge. Choose an item from the list to ne source code for this relationship. You can limit the length of this list as bed for the <b>Tools &gt; Options</b> dialog on page 118.				
	You can customize the display colors, shapes, Graphs category of the <b>Tools &gt; Options</b> dialo	and arrows used in cluster graphs in the og (page 118).				
Graph Customizer Toolbar	The toolbar icons in the Graph Customizer pane perform the following actions:					
	• Save icon. Prompts you for a name for the current settings. Settings apply only to the specific graph type and root node in this view. If you have already saved settings for this graph type/root node combination, you can select a set you want to update from the context menu. Otherwise, type a name for your current settings and click <b>Save</b> .					
	<ul> <li>Load icon. Prompts you to select a named open in the current window. The list shows type/root node combination. To see the full Dependency Graphs &gt; Load Saved Dependency</li> </ul>	group of graph settings that you want to only settings saved for this graph list of saved settings, choose <b>Graphs &gt;</b> endency Graph.				
	• Undo icon. Undo your last change.					
	• Redo icon. Redo the last change you und	id.				
	• Restore Defaults. Restores graph to the s	ettings it had when you opened it.				
	• Selected Node tools. Use this drop-down currently selected nodes and edges. The ic the most recently used command. The com you have selected nodes, whether any sele and whether any selected nodes have child	menu to control the display of the con for this toolbar item changes to reflect mands are active depending on whether acted nodes have edges coming in or out, dren.				
You can select one or more nodes in a dependency graph by using your m drag a rectangle over the nodes you want to select. Or, hold down the Ctrl I clicking on multiple nodes you want to select.						
	<ul> <li>Show Selected Node Children. Causes be displayed. This is the same as double</li> </ul>	s any child nodes of the selected node to e-clicking on a node to expand it.				
	<ul> <li>Show Edges Between Children. Cause selected child node and any other child r display of arrows, the graph is reorganize</li> </ul>	es arrows to be drawn between the nodes as appropriate. If you remove the ed to hide these relationships.				

- Aggregate Child Edges Going Out. Causes arrows coming from the selected node's children to be drawn coming from the node, and arrows with the same target from multiple children to not be repeated. Toggle this off to cause separate arrows to be drawn from the individual child nodes.
- Aggregate Child Edges Coming In. Causes the arrows going to the selected node's children to be drawn as going to the node, and arrows to multiple children are not repeated. Toggle this off to cause separate arrows to be drawn to the individual child nodes.
- Highlight Edges Going Out. Causes the selected node to be highlighted in yellow. Arrows pointing from this node to other nodes become darker. Nodes to which they point are highlighted in light blue. Internal Dependency graphs let you highlight such edges; other types of dependency graphs let you show or hide such edges.



- Highlight Edges Coming In. Causes the selected node to be highlighted in yellow. Any arrows that point to this node from other nodes become darker, and nodes which point to this node are highlighted in light blue. Internal Dependency graphs only let you highlight such edges; other types of dependency graphs let you show or hide such edges.



- Hide Selected Node(s). Removes all the nodes that are currently selected from the graph and reorganizes the graph as needed. (You can later restore the hidden nodes by clicking the Show All Hidden Nodes button.)
- **Global Node tools.** Use this drop-down menu to show and hide various nodes and edges. The icon for this toolbar item changes to reflect the most recently used command.
  - Hide Nodes With No Highlighted Edges. This field is available only for Internal Dependency graphs, and you can use it only if you have turned on highlighting of "edges." If you check this box, all nodes that do not have a highlighted arrow pointing to it or away from it are hidden, and the graph is reorganized as needed to omit those nodes.
  - **Hide Unhighlighted Edges.** This field is available only for Internal Dependency graphs, and you can use it only if you have turned on highlighting of "edges," which are the connections between nodes. If you check this box, all arrows that are not highlighted are hidden, and the graph is reorganized as needed to omit those non-highlighted relationships.
  - Clear All Highlighted Edges. This button is available only for Internal Dependency graphs. If you click this button, all node and "edge" highlighting is removed.

- Show All Hidden Nodes. If you click this button, any nodes that have been hidden using the "Hide Selected Nodes" button are restored. This button does not expand any nodes that have been contracted to hide child nodes. If you have hidden any nodes, you can select entities from the **Hidden Nodes** drop-down list to redisplay those nodes.
- **Dependencies Shown.** You can choose the types of dependencies you want shown in the graph from this list. The dependency types available include Inits (initializes), Sets, Uses, Calls, and Modifies. By default, all types of dependencies are shown. If you have the **Sync** box checked in the Dependency Browser, the current Show settings in the dependency graph are copied to the Dependency Browser when you click on a relationship in the graph.
- **Open Dependency Browser.** This button opens the Dependency Browser (page 140) for the most recently selected node or relationship in the graph. Syncing between the dependency graph and the Dependency Browser is turned on by default when you open the Dependency Browser this way.

#### Graph Architecture View

The **Graph Architecture** view is different from the Architecture Dependency graphs. It shows the structure of the architecture, rather than dependencies between entities in the architecture. Open the Graph Architecture view the same way you would open other graphical views (see page 248).

This type of graph does not provide a Graph Customizer panel, but you can right-click on the graph to modify the display. For example, in the following Architecture Graph, **Include Entity Lists** was off by default but was turned on by right-clicking.



## **Viewing Architecture Metrics**

You can generate metrics information about an architecture or a subset of an architecture. The metrics information can be either a text summary or a comma-separated list for use in spreadsheets.

To create a metrics summary, follow these steps:

- 1 Select the highest-level node of the architecture for which you want metrics.
- 2 Right-click on the node and choose **Metrics Summary** from the context menu.
- 3 You see an Architecture Metrics Summary window.

_					_
P	Ge	tting Started 🗙 🗟 Untitled 1 🗙			μ
1		Language/C C++/C - Arc	chit	tecture Metrics Summary	
2					^
3		AltAvgLineBlank	:	: 99.00000	
4		AltAvgLineCode	:	: 1102.000000	
5		AltAvgLineComment	:	: 287.00000	
6		AltCountLineBlank	:	: 1184.000000	=
7		AltCountLineCode	:	: 7831.000000	
8		AltCountLineComment	:	: 3056.000000	
9		AvgCyclomatic	:	: 10.190000	-
10		AvgCyclomaticModified	:	: 9.820000	
11		AvgCyclomaticStrict	:	: 11.920000	
12		AvgEssential	:	: 5.220000	
13		AvgLine	:	: 543.710000	
14		AvgLineBlank	:	: 50.330000	
15		AvgLineCode	:	: 297.380000	
16		AvgLineComment	:	: 133.620000	
17		CountDeclClass	:	: 0.00000	
18		CountDeclFile	:	: 21.000000	
19		CountDeclFileCode	:	: 21.000000	
20		CountDeclFunction	:	: 173.000000	÷

4 When you close the window, you are asked whether you want to save the file. If you click **Save**, you can save the summary as text.

To create a metrics export file, follow these steps:

- 1 Select the highest-level node of the architecture for which you want metrics.
- 2 Right-click on the node and choose Metrics Export from the context menu.
- 3 You see a comma-separated values file. The heading label for each column is in the first row. Each node in the architecture hierarchy has a separate row with metrics for that node's contents.

E	Un	titled 2×
1		Name, AltAvgLineBlank, AltAvgLineCode, AltAvgLineComment, AltCountLineBlank, AltCo
2		Language/C C++/C++,0,0,0,289,1771,1658,0,0,0,0,293.58,23.92,81,135,0,12,1,11,
3		Language/C C++/C++/zlib,0,0,0,289,1771,1658,0,0,0,0,293.58,23.92,81,135,0,12,
4		Language/C C++/C++/zlib/examples,0,0,0,4,3,51,0,0,0,0,58,4,3,51,0,1,,1,0,58,4
5		

4 When you close the window, you are asked whether you want to save the file. If you click **Save**, you can save the data as a .CSV file.

#### **Managing Architectures**

To open the Architect Manager window, choose **Project > Architectures > Manage Architectures** from the main menu bar in *Understand*. The window lists the autoarchitectures on the right and custom architectures you have created on the left.

<ul> <li>Architect Manager</li> </ul>				9
0 2 / 0 0 1 🔁	1			
Custom Architectures	Status	Date Last Modified	Auto Architectures	Status
Functional Decomposition	Show	Friday, September 06, 2013	Calendar	V Enable
Requirements	Show	Friday, September 06, 2013	Directory Structure	V Enable
test	Show	Friday, September 06, 2013	Language	V Enable

The checkboxes allow you to control whether custom and auto architectures are shown in the Architectures area. Removing the checkmark next to an architecture can improve performance, especially for large projects. So, you might want to disable/hide architectures you never or rarely use.

You can use the icons at the top of this area or right-click on an architecture to perform the following actions:

- Create a new architecture: See page 202.
- **Edit architecture:** Predefined and custom architectures only. See page 204.
- Rename architecture: Predefined and custom architectures only. See page 202.

- Duplicate architecture: See page 202.
- Delete architecture: Predefined and custom architectures only.
- Import architecture from XML: See page 206.
- Export architecture to XML: See page 206.

#### **Creating an Architecture**

There are several ways to create a new architecture:

- To create an architecture from scratch, choose **Project > Architectures > New Architecture** from the menus or click the icon in the Architect Manager. Use the Architecture Wizard to create the architecture as described in *Using the Architecture Wizard* on page 203.
- To duplicate an existing architecture (which you can then modify), select an architecture and click the icon in the Architect Manager window. Or, right-click an existing architecture node in the Architecture Browser and choose **Duplicate Architecture** from the context menu to create an architecture from that node and lower in the hierarchy.

Duplicate Architecture
Please enter a name for the architecture.
OK Cancel

You can rename an architecture you have created by selecting an architecture and clicking the *l* icon in the Architect Manager window. Or, right-click on an existing custom architecture and choose **Rename Architecture** from the context menu.

Using theWhen you open the Architecture Wizard by choosing Project > Architectures > NewArchitecture WizardArchitecture from the menus or clicking the icon in the Architect Manager window,

Architecture from the menus or clicking the **D** icon in the Architect Manager window, you see a page that asks for the name of your architecture.

Type a name for the architecture. This name should be fairly short so it can be shown in architecture trees.

Architecture Wizard		? ×
Create An Architecture		
An architecture is an abstract hierarch	hy layered onto a body of source code.	
For example, a staff architecture couk engineer would contain a list of source information can then be derived from t	d list which engineers worked on a particular project ar e code belonging to them. Dependencies and other use he architecture. First, you need to give the architecture	id each ful a name.
Enter a name for the architecture:		
Cancel	< Back	Next >

Then click **Next** to see the page that lets you add and edit architecture nodes. This is the hierarchy to which entities will be assigned in a later page of the wizard

Architecture Wizard	? ×
Create An Architecture	
An architecture consists of hierarchical nodes or sub-architectures that contain source of sub-architectures. You can add sub-architectures to this architecture or edit it later.	code or other
Components Architecture	Add A Node
Components	Edit Node
Parser	Lait Node
GUI	Remove Node
API	
Cancel < Back	Next >

Click **Add a Node** and type the **Name** of the node you want to add. The default location is within the node you had selected in the Architecture Wizard, but you can select another location in the **Create In** field. Then click **OK**.

🧏 Add A Node	? ×
Name:	
Create In: Components	▼
	OK Cancel

You can modify nodes you have created by selecting a node and clicking **Edit Node**. You can delete the selected node by clicking **Remove Node**.

The next window presents an animation that shows how to use the Architecture Builder to add entities to the nodes you have created. When you have finished watching the animation, click **Finish**. This opens the Architecture Builder shown in the animation. Your architecture nodes are shown on the right. See *Editing an Architecture* on page 204 for details on adding entities to each node.

### **Editing an Architecture**

You can quickly add a file to an existing custom architecture by right-clicking on a file in the Project Browser and choosing **Add to Architecture** and the architecture node within which you want to place the file.

<ul> <li>Project Browser - getopt.udb</li> <li>B</li> <li>C</li> <li>B</li> <li>C</li> <li>C</li></ul>	Edit File	wa <sub>X</sub> 🚱 REApp	let.ja	va 🗶 🔎 Gett	ing Sta	arted×	<u>ا</u> ھ	Untitled 1 <sub>×</sub>
⊿ 📁 regexp ⊿ 📁 util	Edit Companion File(s) Rename File							
<ul> <li>Egrep.java</li> <li>Grep.java</li> </ul>	Add to Favorites							
REApplet.java	Compare •							
Tests.java	Analyze CharlndexedStringBuffer.java							
CharIndexed.java	Remove from Project							
CharlndexedCharArray.java	Add To Architecture	Component +	4	Add to: Comp	onent			
CharindexedinputStream.java	Annotate •		-					
CharindexedReader.java				util	•			
CharindexedString.java	User Tools			Parser	•	+	Add to	: Parser
CharindexedStringBuffer.java	Explore •			Error Handling	i +	-		
RE.java     REferention inve	Find in C:\Program Files\SciTools\sample			Timina	· .			
REFitterInputStream iava	Metrics Charts			Liner Interface				
	Browse Metrics			o ser internaci				
	Copy Full Name		_	Communicatio	ns 🕨			
	<ul> <li>Project Browser - getopt.udb</li> <li>Prile Sync</li> <li>regexp</li> <li>util</li> <li>Egrep.java</li> <li>Grep.java</li> <li>REApplet.java</li> <li>RETest.java</li> <li>Tests.java</li> <li>CharIndexed.java</li> <li>CharIndexedReader.java</li> <li>CharIndexedString.java</li> <li>CharIndexedString.java</li> <li>CharIndexedStringBuffer.java</li> <li>REEjava</li> <li>REException.java</li> <li>REFilterInputStream.java</li> </ul>	<ul> <li>Project Browser - getopt.udb</li> <li>Project Browse Metrics</li> <li>Project Bro</li></ul>	<ul> <li>Project Browser - getopt.udb</li> <li>Project Browser - getopt.udb</li> <li>File Sync</li> <li>File Sync</li> <li>File Sync</li> <li>Edit File</li> <li>Edit Companion File(s)</li> <li>Rename File</li> <li>Add to Favorites</li> <li>Grep.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>RETest.java</li> <li>Tests.java</li> <li>Charlndexed.java</li> <li>CharlndexedCharArray.java</li> <li>CharlndexedReader.java</li> <li>CharlndexedStringBuffer.java</li> <li>CharlndexedStringBuffer.java</li> <li>CharlndexedStringBuffer.java</li> <li>CharlndexedStringBuffer.java</li> <li>RE.java</li> <li>REE.java</li> <li>REE.xception.java</li> <li>REFilterInputStream.java</li> <li>Browse Metrics</li> <li>Copy Full Name</li> </ul>	<ul> <li>Project Browser - getopt.udb</li> <li>Project Browser - getopt.getopt</li></ul>	<ul> <li>Project Browser - getopt.udb</li> <li>Project Browser - getopt.udb</li> <li>Prile Sync</li> <li>Edit File</li> <li>Edit Companion File(s)</li> <li>Rename File</li> <li>Add to Favorites</li> <li>Gerp.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>Gerp.java</li> <li>Gerp.java</li> <li>Grep.java</li> <li>Charindexed.java</li> <li>CharindexedCharArray.java</li> <li>CharindexedReader.java</li> <li>CharindexedStringBuffer.java</li> <li>CharindexedStringBuffer.java</li> <li>CharindexedStringBuffer.java</li> <li>CharindexedStringBuffer.java</li> <li>CharindexedStringBuffer.java</li> <li>RE.java</li> <li>REException.java</li> <li>REFilterInputStream.java</li> <li>REFilterInputStream.java</li> <li>Copy Full Name</li> </ul>	<ul> <li>▶ Project Browser - getopt.udb</li> <li>▶ Project Browser Fries\ScrTools\sample</li> <li>▶ Parser ▶ Error Handling ▶ Timing ▶ User Interface ▶ Communications ▶ Copy Full Name</li> </ul>	<ul> <li>Project Browser - getopt.udb</li> <li>Project Browser - getopt.udb</li> <li>Prile Sync</li> <li>Pregexp</li> <li>Util</li> <li>Egrep.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>Grep.java</li> <li>REApplet.java</li> <li>Charindexed.yava</li> <li>Charindexed.tarArray.java</li> <li>Charindexed.tarAray.java</li> <li>Charindexed.tarArray.java</li> <li></li></ul>	<ul> <li> Project Browser - getopt.udb Edit File Edit File Edit Companion File(s) Rename File Add to Favorites Compare Add to Favorites Compare Analyze CharindexedStringBuffer.java CharindexedCharArray.java CharindexedCharArray.java CharindexedStringBuffer.java CharindexedStringBuffer.java CharindexedStringBuffer.java Setting Started, Se</li></ul>

To make changes to an architecture beyond adding files, select that architecture and click the *click* icon in the Architect Manager window. Or, right-click on an existing architecture and choose **Edit Architecture** from the context menu. Both actions open the Architecture Builder.

You cannot edit the Auto Architectures provided with *Understand*. However, you can use the *i* icon in the Architect Manager window to create a duplicate architecture of one of the Auto Architectures. Then, you can edit the duplicate architecture.

This dialog allows you to add nodes to architectures. You create an architecture structure on the right-hand side and map entities into the architecture from the left-hand side.

Architecture Builder			
Architectures Filesystem  Show All Entities  Show Unmapped Entities  Directory Structure  Pixie  Src  config.windows.h (File)	Add >> Remove	Architecture: Requirements Requirements GUI Changes Memory Use Issues Performance Licensing	
config.xcode.h (File)	J		×

- Multiple items may be selected and mapped to/from both trees.
- Drag and drop can be used to add items from the left to the right or to re-arrange nodes in the right tree.
- Use the radio buttons to toggle between the full architecture and the architecture containing only nodes that are unmapped.

11111111111

To create and edit nodes in the Architecture Builder, follow these steps:

- Double-click the name of any node on the right side of the Architecture Builder, and rename that node by typing. (Or you can select a node and press Enter.)
- Move one or more nodes by dragging them to the node you want to be the parent node. Within a node, the children are sorted alphabetically.
- Click the icon to create a new node at the same level as the selected node.
   Click the icon to create a new node as the child of the selected node.
- Click the 💓 icon to delete the selected node.
- Click the Discrete icon to undo your last change. Click the Circle icon to redo you last undo.

To map files to nodes in the Architecture Builder, follow these steps:

1 On the left side of the Architecture Builder, select an existing architecture from the drop-down list that will allow you to easily find the files you want. The default is the Directory Structure architecture.

- 2 You can choose whether to show all entries in the architecture or just the unmapped entries. For example, if you want to map all the entries into your new architecture, you might want to select **Show Unmapped Entries** so that you can see which files you haven't mapped yet.
- 3 In the left architecture hierarchy, select one or more files or architecture nodes.
- 4 In the right architecture hierarchy, select the node you want to contain your selection.
- 5 Click the Add button or drag your selection to the right side.
- 6 When you finish editing your custom architecture, click Save.

You can use the **Remove** button to delete files and nodes from the architecture you are editing.

As always, you can right-click on any node or file to use its context menu to get information.

You can save your edits to the architecture at any point by clicking the [-] icon. Then, you can continue editing. If you close the Architecture Builder without saving changes, you will be asked if you want to save your changes.

#### **Using XML to Manage Architectures**

You can use XML as a way to share architectures between one *Understand* database and another.

In addition to using XML to share architectures, you can use XML export/import to quickly create architectures that are a simple subset of another architecture by selecting a lower node in the hierarchy.

Exporting	To create an XIML file for an architecture, follow these steps:							
Architectures to XML	1	Select the highest-level node of the architecture that you want to export. All of the hierarchy below the node you select will be represented in the XML file.						
	2	Click the a icon in the Architect Manager window. Or, right-click on the node you selected and choose <b>XML Export</b> from the context menu.						
	3	You see an XML file that contains <arch> and <set> tags for architecture nodes.</set></arch>						
	4	When you close the XML window, you are asked if you want to save the file. If you click <b>Save</b> , the default filename is the name of the node you selected.						
Importing XML	Тс	o import an XML file for an architecture, follow these steps:						
Architectures	1	Click the 📸 icon in the Architect Manager window.						
	2	In the Choose XML File to Import Architecture dialog, select an XML file that matches the tag format used by <i>Understand</i> to describe architectures. For example, you can choose XML files created by <i>Understand</i> . Click <b>Open</b> .						
	3	The architecture described by the XML file is added to your list of architectures.						

# Chapter 8 Using Reports

This chapter describes how to create and view reports and the types of reports available.

This chapter contains the following sections:

Section	Page
Configuring Reports	208
Generating Reports	210
Viewing Reports	211
An Overview of Report Categories	212
Cross-Reference Reports	214
Structure Reports	219
Quality Reports	222
Metrics Reports	226

#### **Configuring Reports**

*Understand* provides a large number of reports you can generate about your code. These can be generated in HTML or text format. You can choose which reports and how to format them.

To configure how reports will be generated, choose **Reports > Configure Reports**. This opens the Project Configuration dialog with the **Reports > Output** category selected. From there, you can also configure the **Reports > Options** and **Reports > Selected** categories.

See page 54 for details on the **Reports > Output** category. In general, you can configure the following:

- **HTML reports:** The "home" file for the reports is index.html, but you can select an alternate title page. You may generate single or multiple HTML files for each report type. It is recommended that you split up the files for large projects. Choose *Alphabetic* to generate multiple HTML files per report that are split up alphabetically by the first letter of the entity name. Choose *Every n Entities* to generate multiple HTML files per report that are split up default, a single HTML file is generated for each letter of the alphabet.
- **Text reports:** You may generate one text file of the specified name (by choosing File). This one file will contain all the selected reports. Alternately, you may generate multiple text files (by choosing Separate Files) and specify a common filename prefix. The filenames of each text file identify the report.

For details on the **Reports > Options** category, see page 55.

The **Reports > Selected** category lets you select from the available reports for the languages used by your project. This list shows all the reports for all languages:



The specific reports available depend upon the source languages used in your project.

See An Overview of Report Categories on page 212 for descriptions of the types of reports you can generate.

Customizing Report Colors	HTML reports use Cascading Style Sheets (CSS) to set colors and font styles used for keywords, comments, strings, numbers, and more. The colors and styles used are defined in the sourcestyles.css file, which is created the first time you generate HTML reports in a particular location.					
	You can customize the sourcestyles.css file using a text editor. Any colors and font styles normally supported by CSS can be used in this file. For example:					
	<pre>span.comment{color:DarkSeaGreen;font-style:italic}</pre>					
	If you modify the stylesheet and want to use if for other reports you generate, you can copy the modified sourcestyles.css file to the locations of other HTML reports.					

#### **Generating Reports**

Once you have specified formatting options and the types of reports to be generated, choose **Reports > Generate Reports** from the menus to open a dialog that lets you begin generating the selected reports. Click **Generate** to show the progress of the report generation.

On Windows, the ASCII text follows the DOS text file format (carriage return and line feed at the end of each line). On Unix, text files are created according to the Unix convention (lines end with a carriage return).

HTML reports are generated as HTML 3.0 format files. The generated HTML is not complex, the only HTML 3.0 (versus HTML 2.0) feature used is frames. Netscape 2.0 and higher, and Internet Explorer 3.0 and higher can display the files.

You can view the reports as described in Viewing Reports on page 211.

For large projects, reports can take a long time to generate. You can click **Cancel** to halt report generation. Clicking **Cancel** leaves the reports in a partially generated state.

- *Note:* You may want to temporarily toggle off anti-virus protection programs while reports are being generated. This may speed the process of creating reports. If you do this, be sure to turn on virus checking after report generation is finished.
- *Note:* HTML, text, and project metrics reports may also be generated with the "und" command line program. Refer to Chapter 14 for details.

#### **Viewing Reports**

To view generated reports, choose **Reports > View Reports**. Then choose the **HTML** or **Text** option.

File names of reports generated vary based on the type and format of the report generated.

- For text files, a single text file containing all selected reports may be generated or separate files for each type of report may be generated. A single text file is named <project\_name>.txt. For separate text files, the file name is the type of report.
- For HTML reports, you can generate either a single HTML files for each report type, or smaller files divided either alphabetically by entity name or in groups of N number of entities. An index file is also generated that contains links to all the other HTML reports generated. The main window page is named index.html.

For HTML reports, a single index file contains an alphabetic list of all entities found in all other generated HTML reports. The entities listed in the index have hyperlinks to the Data Dictionary report for that entity. The entity index file is named entity\_index.html and can be accessed from the "index" link on the main HTML page.

The following figure shows an example of the entity index.

		access::list	
Understand	^	access::size	^
Chaelbtana		action	
		add data in datablock	
Table of Contents		add position when writting offset	
		ADD UNDERLINE ASMFUNC	
(Index)		addpoint	
·		adler	
		adler1	
		adler2	
Data Distignary		adler32	
Data Dictionary		adler32 combine	
File Contents		all read before	
<u>Inc contents</u>		ALLOC	
Program Unit Cross		alloc func	
Reference		allocate	
		allocate new datablock	
Object Cross Reference		AllocMem	
		AMIGA	
Type Cross Reference		answer	
		append	
Macro Cross Reference		APPEND STATUS ADDINZIP	

#### **An Overview of Report Categories**

Understand generates a wide variety of reports. The reports fall into these categories:

- **Cross-Reference** reports show information similar to that in the *Information Browser*, except that all entities are shown together in alphabetic order. See Cross-*Reference Reports* on page 214.
- **Structure** reports show the structure of the analyzed program. See *Structure Reports* on page 219.
- **Quality** reports show areas where code might need to be examined. See *Quality Reports* on page 222.
- **Metrics** reports show basic metrics such as the number of lines of code and comments. See *Metrics Reports* on page 226.

The following table shows the type and page number for each report. Note that the specific reports available depend upon the source languages used in your project.

Report Type	Report Name and Page
Cross-Reference	Data Dictionary Report on page 214
Cross-Reference	File Contents Report on page 215
Cross-Reference	Program Unit Cross-Reference Report on page 215
Cross-Reference	Object Cross-Reference Report on page 216
Cross-Reference	Type Cross-Reference Report on page 216
Cross-Reference	Macro Cross-Reference on page 217
Cross-Reference	Include File Cross-Reference on page 217
Cross-Reference	Exception Cross-Reference Report on page 218
Structure	Declaration Tree on page 219
Structure	Extend Tree on page 220
Structure	Invocation Tree Report on page 220
Structure	Simple Invocation Tree Report on page 220
Structure	Import Report on page 221
Structure	With Tree Report on page 221
Structure	Simple With Tree Report on page 221
Structure	Generic Instantiation Report on page 221
Structure	Renames Report on page 221
Quality	Program Unit Complexity Report on page 222
Quality	Uninitialized Items on page 224
Quality	Unused Objects and Functions on page 224
Quality	Unused Objects Report on page 224
Quality	Unused Types Report on page 225
Quality	Unused Program Units Report on page 225
Quality	Uses Not Needed Report on page 225
Quality	Withs Not Needed Report on page 225

	Report Type	Report Name and Page
	Quality	Implicitly Declared Objects Report on page 223
	Quality	Fortran Extension Usage Report on page 223
	Metrics	Project Metrics Report on page 226
	Metrics	Program Unit Metrics Report on page 228
	Metrics	File Metrics Report on page 228
	Metrics	File Average Metrics Report on page 229
	Metrics	Class Metrics Report on page 227
	Metrics	Class OO Metrics Report on page 227
PERL or C API       The reports included with Onderstand have evolved of common customer requests. However, we recognize to help you develop custom reports we include both F Understand databases.         For details on the PERL interface choose Help > PER the blog and support page on our website. Java API of doc/manuals/java subdirectory of the Understand inst         The Reports > Project Interactive Reports and Gra commands display a list of user-created plugins, whic API. For information about creating plugins, please conscitutions.com/support and the scitools.com/blog also contain messages concerning		a customer requests. However, we recognize that not all needs can be covered. you develop custom reports we include both PERL and C interfaces to and databases. ils on the PERL interface choose <b>Help &gt; PERL API Documentation</b> . Also visit and support page on our website. Java API documentation is provided in the nuals/java subdirectory of the <i>Understand</i> installation. <b>Ports &gt; Project Interactive Reports</b> and <b>Graphs &gt; Project Graphs</b> ads display a list of user-created plugins, which can be created using the Perl information about creating plugins, please contact support@scitools.com. The Support page at scitools.com/support and the SciTools blog at com/blog also contain messages concerning plugins.

## **Cross-Reference Reports**

Cross-Reference reports show information similar to that in the References section of the Information Browser, except that all entities are shown together in alphabetic order. The following table shows the page that describes each type of cross-reference report.

Report N	lame
----------	------

Data Dictionary Report on page 214
Program Unit Cross-Reference Report on page 215
File Contents Report on page 215
Object Cross-Reference Report on page 216
Type Cross-Reference Report on page 216
Class and Interface Cross-Reference on page 216
Macro Cross-Reference on page 217
Include File Cross-Reference on page 217
Exception Cross-Reference Report on page 218

#### Data Dictionary Report

The *Data Dictionary Report* lists all entities alphabetically. Each listing shows the entity name, what kind of entity it is (for example, macro, type, variable, function, include, file, or procedure), along with links to the location where each is declared in the source code.



# Program Unit Cross-<br/>Reference ReportThe Program Unit Cross-Reference Report lists all program units (such as procedures<br/>and functions) analyzed in alphabetic order along with information about what they<br/>return (if anything), what parameters are used, and where they are used by other

program units.

The HTML version offers hyperlinks to the Data Dictionary report entry and to the source code where each reference occurs.





**Object Cross-**The Object Cross-Reference Report lists all objects (Fortran variables, parameters, **Reference Report** macros) in alphabetic order along with declaration and usage references. (Parameter) Declared as: unsigned а Define [gzappend.c, 100] gcd Use [gzappend.c, 104] gcd Modify [gzappend.c, 109] gcd Return [gzappend.c, 117] gcd (Public Object) Declared as: i access::have access Define [zran.c, 71] Set [zran.c, 102] Use [zran.c, 106] addpoint addpoint Modify [zran.c, 125] addpoint The HTML version of this report includes hyperlinks to the Data Dictionary Report and the source code where the reference occurs. This report was previously titled the Class and Interface Cross-Reference Report. Type Cross-Reference The Type Cross-Reference Report lists all declared types in alphabetic order, along Report with their declaration and usage information. The HTML version of the report offers hyperlinks to the Types data dictionary report entry, as well as the source code where the reference occurs. gz header (Typedef) Declared as: struct gz header s Define [zlib.h, 124] zlib.h Type [zlib.h, 126] gz headerp gz header s (Struct) Define [zlib.h, 109] zlib.h Type [zlib.h, 109] gz header **Class and Interface** The Class and Interface Cross-Reference Report lists all declared classes and **Cross-Reference** interfaces in alphabetic order, along with their declaration and usage information. The HTML version of the report includes hyperlinks to the data dictionary report entries, as well as the source code where the reference occurs. Error (Unknown Class) Declared as: Create [<u>REMatch.java, 62</u>] REMatch.clone Create [<u>Grep.java, 273</u>] Grep.processStream <u>Event</u> (Unknown Class) Declared as: Typed [REApplet.java, 198] e Exception (Unknown Class) Declared as: Extend [REException.java, 38] regexp.REException Typed [<u>Grep.java, 236</u>] <u>ex</u>
# Macro Cross-<br/>ReferenceThe Macro Cross-Reference Report lists all macros analyzed in the source code in<br/>alphabetic order along with information about where they are declared and where they<br/>are used. The HTML version offers hyperlinks to the macro's Data Dictionary report<br/>entry and to the source code where each reference occurs.



#### Include File Cross-Reference

. . . . . . . . . . . . . . . .

The Include File Cross-Reference Report lists all include files analyzed in the source code in alphabetic order with information about which files include them. The HTML version offers hyperlinks to the source code where each reference occurs.

#### **Include File Cross Reference** N JK в Ε F G H Μ O Non-Alpha D alloc.h Inactive [zutil.h, 97] zutil.h assert.h Include [zpipe.c, 15] zpipe.c [fitblk.c, 56] Include fitblk.c

#### . . . . . . . . . . . . . . . . **Exception Cross**-**Reference Report**

The Exception Cross-Reference Report documents the declaration and usage of all exceptions. Each declaration and any raises or handles are documented. In the HTML version each raise or handle may be visited in the source, as well as the declaration point of the Exception (if visible).

# **Exception Cross Reference**

Non-Alpha	BCDEFGHI	JKLMNOPQ
stack pack	age.stack empty (Excep	tion)
Declar	e [stack pa.adb, 21]	stack package
Raise	[stack pa.adb, 114]	stack package.pop
Raise	[stack pa.adb, 124]	stack package.top
stack pack	age.stack_full (Except	ion)
stack pack Declar	age.stack full (Except e [stack pa.adb, 22]	ion) stack package
stack pack Declar Raise	age.stack full (Except e [stack pa.adb, 22] [stack pa.adb, 104]	ion) stack package stack package.push

## **Structure Reports**

Structure reports are designed to help you understand the relationships between various entities. The following table shows the page in this chapter that describes each type of structure report.

Report Name and P	ige	
Declaration Tree o	page 219	
Extend Tree on pag	je 220	
nvocation Tree Re	p <i>ort</i> on page 220	
Simple Invocation	Free Report on page 220	
With Tree Report o	ו page 221	
Simple With Tree F	<i>eport</i> on page 221	
Generic Instantiatio	<i>n Report</i> on page 221	
Renames Report o	ר page 221	
<i>Import Report</i> on p	age 221	

.....

#### **Declaration Tree**

The *Declaration Tree* shows the declaration nesting of each program unit analyzed. Each nesting level is indicated by an indent with a vertical bar used to help align your eyes when viewing. Each nesting level is read as "declares". In the HTML version of the report each program unit name is a hyperlink to its entry in the *Program Unit Cross-Reference Report*.

```
Package Body Occupants

| Procedure Get

| Function May_I_Get

| Procedure Drop

| Function May_I_Drop

| Procedure Inventory

| Procedure Go

| Block
```

In the above example, Package Body *Occupants* is the top level program unit. It has declared within it, *Put\_View, Look, Get, May\_I\_Get, Drop, May\_I\_Drop, Inventory, and Go.* Nested within *Go* is an unnamed declare block.

The Declaration Tree report shows a representation of the declaration tree in each file.

#### **Declaration Tree**

```
C:\Program Files\SciTools\sample\getopt\GetoptDemo.java File

| <u>GetoptDemo</u> Public Class

| <u>GetoptDemo.main</u> Public Static Method

C:\Program Files\SciTools\sample\getopt\LongOpt.java File

| <u>getopt.LongOpt</u> Public Class

| <u>LongOpt.LongOpt</u> Public Constructor

| <u>LongOpt.getName</u> Public Method

| <u>LongOpt.getHasArg</u> Public Method
```

Extend Tree	The <i>Extend Tree</i> report shows the nesting of class declarations in the files analyzed. Each nesting level is indicated by an indent with a vertical bar to help align your eyes when viewing. Each nesting level is read as "extends". In the HTML version of the report each class name is a hyperlink to its entry in the <i>Data Dictionary and Interface</i> <i>Cross-Reference Report</i> .
Invocation Tree Report	The Invocation Tree Report shows a textual representation of the invocation tree for each program unit analyzed. The report shows what each program unit calls. Levels are indicated by tabs and are lined up with vertical bars. Each nesting level is read as "calls". The HTML version has hyperlinks to the corresponding Data Dictionary report entries.
	<pre>main   strcpy   strcmp   setmode   fileno   gzdopen     sprintf   gz open     malloc       crc32       <u>strlen</u>       <u>destroy</u>       <u>destroy</u>       <u>destroy</u>       <u>deflateEnd</u>       <u>deflateEnd</u>       <u>inflateEnd</u>       <u>strcpy</u>       <u>inflateInit2</u>       <u>strcpy</u>       <u>inflateInit2</u>       <u>zcalloc</u>         <u>*** Repeated Subtree ***</u></pre>
Simple Invocation Tree Report	The Simple Invocation Tree Report shows the invocation tree to only one level for each program unit that has been analyzed. The invocation level is indicated by an indent and a vertical bar and is read as "calls".
	<pre>main   strcpy   strcmp   setmode   fileno   gzdopen   error   gz uncompress   gz compress   file uncompress   file compress</pre>

	Report entry may be visited from hyperlinks.				
	Report entry may be visited from hyperlinks.				
Report	In the HTML version, the source where it was instantiated and its Data Dictionary				
Generic Instantiation	This report lists each package that was created through instantiation.				
	Package Rename Text_IO				
	Package Body Occupants				
	However, it shows only one level of withs. For example:				
Report	representation of the With Tree for each program unit that is not Withed by another.				
Simple With Iree	The Simple With Tree report is similar to the With Tree report. It shows a textual representation of the With Tree for each program unit that is not Withed by another				
Simple With Tree	The Simple With Tree report is similar to the With Tree report. It shows a taxtual				
	turn Withs IO_Exceptions, System, and Parameters.				
	In the above example, the package body <i>Occupants</i> Withs package <i>Text</i> 10, which in				
	Package Barameters				
	Package IO_Exceptions				
	Package Text_IO				
	Package Rename Text_IO				
	Package Body Occupants				
	align your eye. For this report, each line is read as "withs".				
	As with the other textual hierarchy reports, indents show level with a vertical bar helping align your ever. For this report, each line is read as "Withs"				
	textual version of the with the for each program unit that is not writted by another.				
man nee neport	Subclured identically to the other meral city reports, the <i>while the</i> report shows a				
With Tree Report	Structured identically to the other hierarchy reports, the With Tree report shows a				

# **Quality Reports**

Understand's quality reports are designed to provide information about areas of the analyzed source that might not meet standards or that hold the potential for trouble. They also identify areas where extra programming has been done but not needed. This sometimes identifies areas that aren't yet complete, or that haven't been maintained completely.

The following table shows the page in this chapter that describes each type of quality report.

	Report Name and Page
	Program Unit Complexity Report on page 222
	Fortran Extension Usage Report on page 223
	Implicitly Declared Objects Report on page 223
	Uninitialized Items on page 224
	Unused Objects and Functions on page 224
	Unused Objects Report on page 224
	Unused Types Report on page 225
	Unused Program Units Report on page 225
	Uses Not Needed Report on page 225
	Withs Not Needed Report on page 225
	The complete list of quality metrics available in Understand changes frequently - more
	frequently than this manual is reprinted. A complete and accurate list is always available on our website: scitools.com/support/metrics_list/.
Program Unit Complexity Report	frequently than this manual is reprinted. A complete and accurate list is always available on our website: scitools.com/support/metrics_list/. The <i>Program Unit Complexity Report</i> lists every procedure and function or similar program unit in alphabetic order along with the McCabe (Cyclomatic) complexity value for the code implementing that program unit.
Program Unit Complexity Report	frequently than this manual is reprinted. A complete and accurate list is always available on our website: scitools.com/support/metrics_list/. The <i>Program Unit Complexity Report</i> lists every procedure and function or similar program unit in alphabetic order along with the McCabe (Cyclomatic) complexity value for the code implementing that program unit. The Cyclomatic complexity is the number of independent paths through a module. The higher this metric the more likely a program unit is to be difficult to test and maintain without error.
Program Unit Complexity Report	frequently than this manual is reprinted. A complete and accurate list is always available on our website: scitools.com/support/metrics_list/. The <i>Program Unit Complexity Report</i> lists every procedure and function or similar program unit in alphabetic order along with the McCabe (Cyclomatic) complexity value for the code implementing that program unit. The Cyclomatic complexity is the number of independent paths through a module. The higher this metric the more likely a program unit is to be difficult to test and maintain without error. The Modified column shows the cyclomatic complexity except that each case statement is not counted; the entire switch counts as 1.

The Nesting column shows the maximum nesting level of control constructs in this program unit.



#### **Fortran Extension**

This report lists anywhere your source code has non-standard Fortran extensions. The

**Usage Report** report factors in what variant (F77, F90, F95) you chose on your project configuration.

The following is a snippet from a sample Fortran Extension Usage report:



Implicitly DeclaredThe Implicitly Declared Objects Report lists any variables or parameters that wereObjects Reportimplicitly declared using Fortran's implicit declaration mode. Using implicitly declared<br/>variables is considered a risky practice, and this report helps you weed out where the<br/>practice is occurring in your code.

The HTML version offers hyperlinks to the function's Data Dictionary report entry.

Uninitialized Items	The <i>Uninitialized Items</i> report lists items such as variables that are not initialized in the code. The report is organized by file. Each uninitialized item within the file is listed by name along with the line number on which the item is declared. The HTML version offers hyperlinks to the location where the item is declared.
Unused Objects and Functions	The <i>Unused Objects and Functions</i> report lists items that are declared (and perhaps initialized) but never referenced other than that. The report is organized by file. Each unused item is listed by name along with the type of item and the line number on which the item is declared. The function or similar container is shown after the list of unused items within it. Types of items may include functions, parameters, variables, and objects. The HTML version offers hyperlinks to the location where each unused item is declared.
Unused Objects Report	The Unused Objects Report lists objects (for example, variables, parameters, constants) that are declared but never used. The HTML version has links to the function's Data Dictionary report entry and to the source line where the object is declared.

Unused Objects	
Non-Alpha A B C D E F	<u>G</u> H <u>I</u> JKLMNOPQR
inffixed.h distfix lenfix	87 10
<u>inftrees.c</u> <u>inflate copyright</u>	11

## **Unused Types Report**

The Unused Types Report lists types that are declared but never used. The HTML version has links to the function's Data Dictionary report entry and the source where the type is declared.



#### **Unused Program Units Report**

The Unused Program Units Report identifies program units that are declared but never used.

Note that this listing in this report doesn't mean the system doesn't need this program unit. For instance, interrupt handlers that are called by system interrupts are often never "used" within the other source of the program.

	Unused Program Units							
	$\boxed{ Non-Alpha \underline{A} \underline{B} \underline{C} \underline{D} \underline{E} \underline{F} \underline{G} \underline{H} \underline{I} \underline{J} \underline{K} \underline{L} \underline{M} \underline{N} \underline{O} \underline{P} \underline{Q} \underline{R} }$							
	deflate.c       deflateBound     489       deflateCopy     894       deflateSetHeader     393       deflateTune     454							
Uses Not Needed Report	The Uses Not Needed Report identifies any unneeded "use" statements that provide access to a module's public specifications and definitions. To remove unneeded access, you may add only clauses to use statements.							
Withs Not Needed Report	This report lists, any With statements a program unit has but does not need (by not using items made public by the With statement).							
	Note that this covers only direct usage in the program unit and doesn't account for side effects that may be needed by the program to operate correctly. For instance, sometimes a package can be Withed just to start a task or to execute code in its begin/end block.							

## **Metrics Reports**

Metrics provide statistical information about your project and entities, such as the number of lines of code and the complexity of various entities.

*Understand* provides a number of ways to gather metrics information. This section describes reports that provide metrics. See page 231 for other ways to gather metrics.

The following table shows the page in this chapter that describes each type of metrics report.

The complete list of metrics available in *Understand* changes frequently—more frequently than this manual is reprinted.

A complete and accurate list is always available on our website: scitools.com/support/metrics\_list/.

#### **Project Metrics**

Report

The *Project Metrics Report* provides metric information about the entire project. The metrics reported include: the total number of files, the total number of program units, and the total number of lines of source code.

## **Project Metrics**

Classes:	0	
Files:	33	
Program Units:	173	
Lines:	14941	
Lines Blank:	1344	
Lines Code:	7217	
Lines Comment:	4426	
Lines Inactive:	1623	
Executable Statements:	5152	
Declarative Statements::	1074	
Ratio Comment/Code:	0.61	

These metrics are also reported on the title page of the HTML report.

#### .....

**Class Metrics Report** The *Class Metrics Report* provides the following metrics for each class that has been analyzed:

- Total number of lines
- Total number of blank lines
- Total number of lines of code
- Total number of lines that contain comments
- Average number of lines per class
- Average number of comment lines per class
- Average complexity per class
- Maximum complexity within class
- Ratio of comment lines to code lines

#### Class Metrics

Class	Lines	<u>Lines</u> <u>Blank</u>	Lines Code	Lines Comment	Average Lines	Average Lines Comment	Average Complexity	Maximum Complexity	<u>Ratio</u> Comment/Code
GetoptDemo	85	2	70	15	79	14	16	16	0.21%
gnu.getopt.Getopt	960	125	506	337	43	5	6	45	0.67%
gnu.getopt.LongOpt	153	25	48	81	6	0	1	2	1.69%

#### Class OO Metrics Report

The *Class OO Metrics Report* provides the following object-oriented metrics for each class that has been analyzed:

- LCOM (Percent Lack of Cohesion): 100% minus the average cohesion for class data members. A method is cohesive when it performs a single task.
- DIT (Max Inheritance Tree): Maximum depth of the class in the inheritance tree.
- IFANIN (Count of Base Classes): Number of immediate base classes.
- CBO (Count of Coupled Classes): Number of other classes coupled to this class.
- NOC (Count of Derived Classes): Number of immediate subclasses this class has.
- RFC (Count of All Methods): Number of methods this class has, including inherited methods.
- NIM (Count of Instance Methods): Number of instance methods this class has.
- NIV (Count of Instance Variables): Number of instance variables this class has.
- WMC (Count of Methods): Number of local methods this class has.

	•••••••••••••••••••••••••••••••••••••••
<b>Program Unit Metrics</b>	The Program Unit Metrics Report provides information on various metrics for each
Report	program unit that has been analyzed.

## **Program Unit Metrics**

	<u>Lines</u>	Comments	<u>Blanks</u>	<u>Code</u>	Lines- exe	Lines- dec	Stmt- exe	Stmt- dec	Ratio Comment/Code
uncompress	36	1	6	29	19	7	22	2	0.03
updatewindow	49	3	4	42	29	6	29	2	0.07

The following metrics are provided for each program unit:

- Lines: Total number of lines in the function.
- **Comment:** Number of comment lines in the function.
- Blank: Number of blank lines in the function.
- Code: Number of lines in the function that contain any code.
- Lines-exe: Lines of code in the function that contain no declaration.
- Lines-decl: Lines of code in the function that contain a declaration or part of a declaration.
- Stmt-exe: Number of executable statements in the function.
- **Stmt-decl:** Number of declarative statements in the function. This includes statements that declare classes, structs, unions, typedefs, and enums.
- Ratio Comment/Code: Ratio of comment lines to code lines. (comment\_lines/code\_lines)

*Note:* code+comment+blank != lines because some lines contain both code and comments.

 File Metrics Report
 The File Metrics Report provides information similar to that in the Program Unit Metrics Report. However, it is organized by file rather than by program unit.

 Click on each metric column to get a detailed description of it.

 Note:
 code+comment+blank != lines because some lines contain both code and comments.

## File Metrics

	<u>Lines</u>	Comments	<u>Blanks</u>	<u>Code</u>	Lines- exe	<u>Lines-</u> <u>dec</u>	<u>Stmt-</u> exe	<u>Stmt-</u> <u>dec</u>	Ratio Comment/Code	<u>Units</u>
deflate.c	1736	456	181	916	617	181	598	109	0.50	22
deflate.h	331	170	61	88	0	88	0	81	1.93	0

File Average Metrics Report	The <i>File Average Metrics Report</i> provides averages for the functions within a file. All lines outside any function are ignored when calculating the averages. The following metrics are provided for each function:
	• <b>Cyclomatic:</b> The average number of independent paths through the functions in this file. The higher this metric the more likely a program unit is to be difficult to test and maintain without error.
	• <b>Modified:</b> Same as Cyclomatic complexity except that each case statement is not counted; the entire switch statement counts as 1.
	• Strict: Same as Cyclomatic complexity except that && and    also count as 1.
	• Essential: Measures the amount of unstructured code in a function.
	Lines: Average number of lines in the functions in this file.
	• Code: Average number of lines that contain any code in the functions in this file.
	• Comment: Average number of comment lines in the functions in this file.
	• Blank: Average number of blank lines in the functions in this file.

# Chapter 9 Using Metrics

This chapter describes how to create and view metrics and the types of metrics available.

This chapter contains the following sections:

Section	Page
About Metrics	231
Metrics Summary	232
Metrics Browser	233
Exporting Metrics to HTML	234
Exporting Metrics to a CSV File	235
Configuring Metric Charts	237
Using the Metrics Treemap	240
Exporting Dependency Metrics	242

## **About Metrics**

Understand provides a number of ways to gather metrics information:

- **Information Browser:** The Information Browser tree has a **Metrics** node. You can expand this branch to show a few metrics for the current entity. See page 135.
- Metrics Summary: You can choose Metrics > Metrics Summary from the menus to see a short list of metrics for the entire project. See page 232.
- Metrics Browser: You can choose Metrics > Browse Metrics from the menus to see a browser that lets you choose any architecture node, file, or entity to see all the metrics available for that item. See page 233.
- **Export to HTML:** You can click this button in the Project Metrics Browser to export the full list of metrics for all architecture nodes and files. See page 234.
- Export to CSV: You can choose Metrics > Export Metrics from the menus to create a text file of all the project metrics in comma-delimited format. See page 235. (You can schedule this export to occur regularly; see page 50.)
- **Configure Metric Charts:** You can choose **Metrics > Configure Metric Charts** from the menus to open a dialog that lets you display graphs of volume and complexity metrics on an architecture basis. See page 237.
- Reports: When you create reports by choosing Project > Project Reports, some of the reports provide metrics. See page 226.
- **PERL/C API:** A more advanced way to get existing metrics and calculate new metrics is with the PERL and C API. These provide full access to the *Understand* database. Choose **Help > PERL API Documentation** for more information. See page 213.

*Understand* provides a large number of metrics you can generate about your code. The complete list of metrics available in *Understand* changes frequently—more frequently than this manual is reprinted.

A complete and accurate list of metrics is always available on our website at scitools.com/support/metrics\_list/. The "What do the metric names mean?" buttons in metrics dialogs links to this page.

# **Metrics Summary**

Choose **Metrics > Metrics Summary** from the menus to see a short list of metrics for the entire project.

Metrics Summary	? ×
Metrics Summary	Copy All
Metric	Value
Blank Lines	3,271
Classes	30
Code Lines	15,167
Comment Lines	8,162
Comment to Code Ratio	0.54
Declarative Statements	2,807
Executable Statements	9,336
Files	82
Functions	497
Inactive Lines	3,268
Lines	30,083
Preprocessor Lines	2,275
Subprograms	45
What do the metric name	s mean? Close

If your cursor hovers over a metric, you will see a short description of that metric. A complete and accurate list of metrics is always available on our website at scitools.com/support/metrics\_list/. The "What do the metric names mean?" buttons in metrics dialogs link to this page.

## **Metrics Browser**

To open the Project Metrics Browser, choose **Metrics > Browse Metrics** from the menus. You can also open the Metrics Browser with a particular architecture node, file, or entity selected by right-clicking on the name of an item in *Understand* and selecting **Browse Metrics**.

You can browse architectures in your project and select any architecture node, file, or entity. The list on the right shows code size and complexity metrics for the selected item. If your cursor hovers over a metric, you will see a short description of that metric. For a full list of metrics with descriptions, see scitools.com/support/metrics\_list/. The "What do the metric names mean?" button links to that page.



If you check the **Sync** checkbox, clicking on an entity elsewhere in *Understand* causes the Metrics Browser to display the metrics for that entity.

Double-click a file or entity to open the Source Editor for that item. Right-click to see the standard informational menu choices.

You can select rows on the right and click **Copy Selected** or press Ctrl+C to copy those lines to the clipboard. Click **Copy All** to copy the full list of metrics for the selected directory or file.

Click **Export to HTML** to generate reports as described on page 234. You can choose to **Export All Architectures** and all of their nodes or to **Export Selected Architectures** and nodes that you have selected using the Ctrl or Shift key while clicking on the nodes you want in the metrics export. Click **Generate Detailed Metrics** to open the Export Metrics dialog and generate a text file in comma-delimited format as described on page 235.



## **Exporting Metrics to HTML**

You can click the **Export to HTML** button in the Project Metrics Browser (page 233) to export the full list of metrics for all entities and architecture nodes. When you click this button, you see a Browse for Folder dialog appropriate to your operating system.

Choose or create the folder where you want the metrics files to be created. The files are actually stored in a folder called "pixie\_proj\_Metrics" below the folder you select.

If the directory already exists, you are asked if the files should be overwritten. If you answer "No", a number is appended to the old directory name to it to save it as a backup.

If the report is generated successfully, you are asked if you want to view the report. Click **Yes** to open the top-level page, index.html.

Architecture:	Cale <sub>l</sub>		
<u>Summary</u>		File: crc32.c	
File Types		Metrics	
C Header File 11 C Code File 22		AltCountLineCode MaxCyclomaticStrict	287 7
Entity Kinds		AvgCyclomatic CountLine	5 423
None		AltAvgLineBlank CountLinePreprocessor	3 64
<u>Metrics</u>		RatioCommentToCode CountLineBlank	0.41 42
CountLineComment	4,426	MaxCyclomaticModified	7
CountLineInactive	1,623	AvgLineCode	21
CountLinePreprocessor	1,599	AvgCyclomaticModified	5
RatioCommentToCode	0.61	CountLineInactive	142
CountDeclFile	33	AltAvgLineComment	2
CountLineBlank	1,344	AvgEssential	1.86
CountStmtDecl	1,074	CountDeclFunction	7
CountDeclFunction	173	AltAvgLineCode	22
CountLine	14,94	AvgLine	26
CountStmtExe	5,152	CountStmtEmpty	0
CountLineCode	7,217	CountStmtExe	128
CountDeclClass	0	CountStmt	151
		AltCountLineBlank	53

The HTML-based report lets you select any architecture node, file, function, or other entity type that has metrics in the left pane. The right pane shows metrics available for that item.

# **Exporting Metrics to a CSV File**

You can save metric information to a comma-delimited text file by choosing **Metrics** > **Export Metrics** from the menus or clicking the **Generate Detailed Metrics** button in the Project Metrics Browser. You can use the generated file in Excel and other spreadsheet programs. The Export Metrics dialog looks like this:

Export Metrics	? ×	
Available Metrics	Metrics To Export	
SELECT ALL AltAvgLineBlank AltAvgLineCode AltAvgLineComment AltCountLineBlank AltCountLineCode AltCountLineCode AltCountLineComment AltCountLineCo	AvgCyclomatic         AvgCyclomaticModified         AvgCyclomaticStrict         CyclomaticModified         CyclomaticStrict         MaxCyclomaticModified         MaxCyclomaticModified         MaxCyclomaticModified         MaxCyclomaticStrict	
Save output to: C:\ProgramData\zlib.csv		
Show File Entity Names As: Short  Show Declared in File: Short	Write Column Titles	
·	View File After Export Close	

The defaults in this dialog come from the Project Configuration dialog in the **Metrics > Options** category (page 52) and the **Metrics > Selected** category (page 53).

You can override the defaults using the following fields:

- Available Metrics: Check the boxes next to metrics you want to include in the output. Check the "SELECT ALL" box to select all metrics. Uncheck the "SELECT ALL" box to unselect all metrics. If your cursor hovers over a metric, you will see a short description of that metric.
- Metrics to Export: Click the single arrow to move the selected metric up or down one in the list. Click the double arrow to move the selected metric to the top or bottom of the list.
- Save output to: Specify the location and name of the file you want to use for metrics output. Understand sends its metrics output to a \*.csv (comma-separated values) file.

- Show File Entity Name as: Specify whether files should be displayed with Short names (just the filename), Full names (including the absolute path), or Relative names (relative directory path).
- Show Declared in File: Check this box if you want the file in which each entity is declared to be included in the output. You can specify whether you want these files displayed with Short names, Full names, or Relative names.
- Write Column Titles: Check this box if you want column headings in the CSV file.
- Show Function Parameter Types: Check this box if you want the type of each function parameter listed.

After setting options, click **Export** to export the .CSV file. If you check the **View File After Export** box before exporting the file, the CSV file is opened with the default application for working with CSV files. This is likely to be a spreadsheet application.

If the output file already exists, you are asked if the files should be overwritten. If you answer "No", you can change the output filename and click **Export** again.

You can schedule this metrics to be automatically exported to a CSV file on a regular basis. See page 50 for details.

See *Exporting Dependency Metrics* on page 242 for more types of metrics you can export to a CSV file.

A complete and accurate list of the available metrics is available at: scitools.com/support/metrics\_list/.

## **Configuring Metric Charts**

Commands in the Metrics menu provide fast access to metrics charts for the current version of the entire project. These commands are:

- Metrics > Project Metric Charts > Code Volume
- Metrics > Project Metric Charts > File Volume
- Metrics > Project Metric Charts > Average Complexity
- Metrics > Project Metric Charts > Sum Complexity

You can choose **Metrics > Configure Metric Charts** from the menus to open the Metric Browser, which lets you display graphs of various metrics on an architecture basis.

In this browser, select the following:

- Architectures: Checkboxes next to one or more architecture nodes, files, and/or entities. The graph will provide a set of vertical bars for each of the items you select.
- **Metrics:** Select the type of metrics you want to graph from the drop-down list.
  - **Code Volume:** Provides a stacked vertical bar chart showing the count of lines that are blank, contain declarations and executable code, and contain comments.
  - File Volume: Provides a vertical bar chart showing the number of files in the selected architecture node that are code files vs. header files (or the number of

P Metric Browser
Architectures          Architectures         Calendar         Directory Structure         Language         C C++         C         fastgrep         C++         fastgrep         regexp.h (File)         regmagic.h (File)
Metic Graph Type
Ode Volume
C File Volume
Average Complexity
Sum Complexity
🕅 Open in a new tab.
View Chart View Table Close

files for languages that do not have header files).

- Average Complexity: Provides a vertical bar chart of the average and maximum cyclomatic complexity for all nested functions or methods in the architecture node, along with the maximum nesting level of control constructs in the node's files.
- **Sum Complexity:** Provides a vertical bar chart showing the number of possible paths through the code and the sum of the cyclomatic complexity and the essential complexity of all nested functions or methods.

Click the **View Chart** button to display a chart for your selections or **View Table** to see the values in a table. If you have already selected a graph for this type of metrics, that tab will be reused unless you check the **Open in a new tab** box.



You can open metrics charts for various entities, including files and functions, from the context menus throughout *Understand*.

A typical metrics chart looks similar to the following:



In a graph, you can choose the **Graph View** or the **Table View**. Both views have a toolbar that lets you save the graph or data.

In the Graph View, you can use the toolbar to:



- Save the image as a PNG, JPEG, or BMP image.
- Copy the image to the clipboard.
- Print the image using the standard Print dialog.

In the **Table View**, the numeric values for each pie slice or vertical bar is shown in a table. You can use the toolbar to copy the data to the clipboard in comma-separated (CSV), tab-separated, or table format (spaces used so columns align with headings if a font such as Courier is used).



If you are viewing data for several architecture nodes, you can change the number in the lower-right corner to the number of vertical bars you want to view on each page and click the checkmark icon. Then use the arrows in the corners to move from page to page. The text shows which vertical bars are currently shown out of the total number. For example, the figure below indicates that bars 3 and 4 out of a total of 5 are currently shown.



## **Using the Metrics Treemap**

Treemaps show metrics graphically by varying the size of node blocks and the color gradient. Each node block represents a code file. Different metrics can be tied to size and color to help you visualize aspects of the code.

For example, the following treemap ties the number of lines in each file to the size of the block and the MaxCyclomatic complexity metric to the darkness of the blue. This allows you to learn which files are large and complex vs. files that are large and relatively non-complex.



So we learn that unzip.c is large, but not particularly complex, while inflate.c is large and highly complex.

By default the maps are nested by directory structure. If you have built other architectures, you can use those as well.

To open the treemap for your project, follow these steps:

1 Choose **Metrics > Metrics Treemap** from the menus. You will see the Metrics Treemap Options dialog.

Metrics Treemap Options
Map metrics for: File
Group by: Directory Structure
Size Options
Map Size to: CountLine
Limit the size of large nodes to 100 👻 % of the available space
Color Options
Map Color to: MaxCyclomatic
Min Color: Max Color:
Use Logarithmic Scale
Cushion Restore Colors
Generate Treemap Cancel

- 2 In the **Map metrics for** field, choose whether you want to select from metrics for Files, Classes/Interfaces/Structs, or Functions/Methods.
- 3 In the **Group by** field, choose how to group blocks in the treemap. The default is to group by the project's directory structure. Alternately, you can choose to group according to any other defined architecture or no architecture (flat).
- 4 In the **Size Options** area, choose a metric to control the size of the blocks. You can also limit the size of the largest blocks to some percentage of the treemap. You might want to use this if one node is taking up so much of the map that you can't see differences between the smaller nodes.
- 5 In the **Color Options** area, choose a metric to control block colors. You can click the left color square to set a color for blocks with the lowest value for this metric; click the right color square to set a color for blocks with the highest value for this metric.
- 6 Check the **Use Logarithmic Scale** box if you want the color scaled by powers of 10 of the selected metric value. This is useful for treemaps with extreme value ranges.
- 7 Uncheck the **Cushion** box if you want to see solid colors in the blocks. By default, the blocks have a gradient fill.
- 8 Click **Generate Treemap** to display the treemap. You can return to the Options dialog by clicking **Options** in the upper-right corner of the treemap.

Within the treemap, when your mouse cursor hovers over a block, the two metric values chosen for the size and color are shown.

You can double-click on an architecture node (shown as a gray border around a set of colored blocks) to display only the contents of that node. You can also zoom in by right-clicking on a node and choosing **Drill down** from the context menu.

After drilling down in the architecture, you can use the  $\bigwedge$  icons to **Pop up one level** or **Pop up all levels** the treemap. You can also right-click to use the **Pop up one level** and **Pop up all levels** commands in the context menu.

The **Print** icon lets you print the treemap.

## **Exporting Dependency Metrics**

The **Reports > Dependency** menu lets you export several types of files that provide metrics about dependencies between architectures, files, and classes/packages.

The output is for most of these commands is in CSV (comma-separated values) format, which can be opened with most spreadsheet programs. When you create a CSV file with *Understand*, it is automatically opened in a text file window.

The options available are as follows:

- Architecture Dependencies >
  - **Export CSV:** This output lists pairs of architecture nodes for which the node in column A is dependent upon the node in column B. The number of dependencies for each pair is listed in column C. (See page 243.)
  - **Export Matrix CSV:** This output lists all architecture nodes that are dependent upon others in column A. Row 1 lists all architecture nodes that are depended upon. The number of dependencies for each pair is listed at the appropriate row/column intersection. (See page 244.)
  - **Export Cytoscape XML:** This output format can be opened in Cytoscape (www.cytoscape.org), a free open-source program for analysis and visualization. It draws large diagrams very quickly, and can be useful if you want an overview picture of dependencies in a very large project. (See page 244.)
  - **Export Dot:** This output format uses a plain text graph description language. Various programs are available that can import DOT files to create graphs, perform calculations, and manipulate the data.
- File Dependencies >
  - **Export CSV:** This output lists pairs of files for which the file in column A is dependent upon the file in column B. The number of dependencies for each pair is listed in column C. (See page 243.)
  - **Export Matrix CSV:** This output lists all files that are dependent upon others in column A. Row 1 lists all files that are depended upon. The number of dependencies for each pair is listed at the appropriate row/column intersection. (See page 244.)

- **Export Cytoscape XML:** See the description of the Cytoscape XML export for architecture dependencies. (See page 244.)
- Class Dependencies >
  - **Export CSV:** This output lists pairs of classes and packages for which the entity in column A is dependent upon the entity in column B. The number of dependencies for each pair is listed in column C. (See page 243.)
  - **Export Matrix CSV:** This output lists all classes and packages that are dependent upon others in column A. Row 1 lists all classes and packages that are depended upon. The number of dependencies for each pair is listed at the appropriate row/column intersection. (See page 244.)
  - **Export Cytoscape XML:** See the description of the Cytoscape XML export for architecture dependencies. (See page 244.)

Exporting Dependencies to a CSV File

When you choose to export a CSV file, you can also set the following options. (This figure shows the dialog for exporting File Dependencies; the other two CSV Export dialogs are very similar.)

File Dependencies Export Options	? <b>x</b>
Output CSV File Path Output File: C:/Users/Yvonne\fastgre	p_FileDependencies.csv
Columns From File From Entities To File To Entities References	Sort      Sort on "From File" column      Sort on "To File" column      Dependency Aggregation
File Names <ul> <li>Short Name</li> <li>Relative Name</li> <li>Long Name</li> </ul>	<ul> <li>Show individual dependency pairs</li> <li>Show sum dependencies for each "From File"</li> <li>Show sum dependencies for each "To File"</li> </ul>
	OK Cancel

In this dialog, you can set the following options:

- Select an architecture to analyze: This option is available only when you are exporting architecture dependencies.
- Output File: Browse for the location to save the CSV file.
- **Columns:** Check the boxes for columns you want to include in the output. The "From" and "To" columns for the type of entity you are exporting are required, and cannot be deselected.

- **Names:** Choose a length for entity names. For example, all types can have a short or long name. Files can also have a relative name.
- Sort: Choose how you want dependencies sorted. You can sort based on the "From" column or the "To" column.
- **Dependency Aggregation:** Choose how you want to summarize dependency pairs that occur multiple times. You can show each pair individually or sum pairs for the "From" or "To" column for the type of entity you are exporting.

#### Exporting Dependencies to a CSV Matrix File

When you choose to export a CSV matrix file, you can also set the following options. (This figure shows the dialog for exporting File Dependencies; the other two CSV Export dialogs are very similar.)

🔎 Dialog		In success	? ×
Output CSV	File Path		
Output File:	C:/Users/Yvonne\fastgrep	p_FileDependencyMatrix.csv	<ul> <li>Browse</li> </ul>
File Names			
Short Na	ime		
Relative	Name		
Long Na	me		
		ок	Cancel

In this dialog, you can set the following options:

- Select an architecture to analyze: This option is available only when you are exporting architecture dependencies.
- Output File: Browse for the location to save the CSV matrix file.
- **Names:** Choose a length for entity names. For example, all types can have a short or long name. Files can also have a relative name.

Exporting Dependencies to	Cytoscape (www.cytoscape.org) is an open source software tool for visualizing complex networks.
Cytoscape	When you choose to export a Cytoscape file, you can Browse to select the location and filename for the output file. If you are exporting architecture dependencies, you can also select an architecture to analyze.
	Once you have exported the *.xml file, you are asked if you want to open the file in Cytoscape. Note that you can only open Cytoscape if it is installed on your computer. See <i>Dependency Category</i> on page 108 for how to configure the location of the Cytoscape installation.

# Chapter 10 Using Graphical Views

This chapter covers the graphical views in Understand and their options.

This chapter contains the following sections:

Section	Page
Project Overview Graphics	246
Graphical View Browsers	248
Types of Views	254
Graphical Notation	263
Controlling Graphical View Layout	263
Controlling Cluster Graph Layout	272
Saving Graphical Views	275
Printing Graphical Views	278

# **Project Overview Graphics**

You can create graphics that provide an overview of your entire project by choosing **Project > Project Overview Charts** from the menus. This opens a tab in the document area that contains a number of pie charts and vertical bar charts. For example:



The graphs provided are code breakdown (line types), function breakdown, class breakdown, most complex functions, largest non-file entities, largest files, largest functions, best comment-to-code ratio entities, and most complex files.

In each area, you can choose the Graph View or the Table View. Both views have a toolbar that lets you save the graph or data.

In the Graph View, you can use the toolbar to:



- Save the image as a PNG, JPEG, or BMP image.
- Copy the image to the clipboard.

- Print the image using the standard Print dialog.
- Zoom in on the graph in a new tab.

In the Table view, the numeric values for each pie slice or vertical bar is shown in a table. You can use the toolbar to copy the data to the clipboard in comma-separated (CSV), tab-separated, or table format (spaces used so columns align with headings if a font such as Courier is used).



In addition to the Project Overview Charts, you can display metrics graphs that provide additional statistical information about your project or portions of your project. For details, see page 237.

## **Graphical View Browsers**

The context menu of an entity that has a structure or hierarchy offers a choice called **Graphical Views**:



You can also use the Graphs drop-down menu in the toolbar to select from the types of graphs available for the entity at the current cursor position in a Source Editor tab. The same list of graphical views is available by choosing **Graphs > Graphs for** *<current\_entity>* from the menus.

The **Graphical Views** menu adapts based on the kind of entity right-clicked. An item is grayed-out if information is normally available for this kind of entity but is not applicable to this particular entity (for instance a package that could be WITHed but isn't).

There are two main types of graphical views in these menus: hierarchy views and structure views.

(Dependency graphs are a separate type of graph described in the chapter on architectures. See page 196 for details. Project overview graphs are described on page 246.)



#### Hierarchy Views

A *hierarchy view* shows multiple level relationships between entities. All relationships are multi-level and are shown to the top or bottom of their respective tree unless a level option is set in the preferences. The following is a Call By graph for a function.



See Hierarchy View Types on page 254 and Hierarchy View Examples on page 255.

Cluster views are a special type of hierarchy view. They provide a more interactive view of call relationships. The Call, Callby, Butterfly and Internal Call variants are available, and can be accessed from the function, class, file, or architecture level. See *Controlling Cluster Graph Layout* on page 272.



**Structure Views** Structure views offer a one glance way to see

Structure views offer a one glance way to see important structure and relational information about a given entity. The following is an example of a Declaration structure view:

🚴 Declaration	n Graph : intersect 🗙
🗊 💩 🔍	🔻 🖑 💽 🛤 🚖 🖛 Export 💌 🗖 Reuse 🗖 Sync
	Parameters:
	CShadingContext * context
	CRay * cRay
intersect returns: void	Calls:
	osLock     processDelayedInstance     osUnlock

See Structure View Types on page 258 and Structure View Examples on page 258.

General Commands for Using Graphical	There are some general commands that can be used for browsing graphical views. Note that some of these tools are not available in all types of graphs.
Browsers	• Entity info: Anywhere you see an entity, you can right-click on it to see a menu that offers many ways to learn more about that entity. Single-clicking shows information about the entity in the Information Browser. If you are in Screen Drag mode or Zoom mode, click the right icon to be able to select entities.
	• Searching: Click the M Search icon at the top of a graphical view or press Ctrl+F to display the incremental search bar. You can use this bar that same way you use it in the Source Editor to find entities by name or other text in the current graphical view. As you type search text, all instances of the string are highlighted in the graphical view. See page 163 for details.
	<ul> <li>Opening source: Right-click on an entity in a graphical view and choose Edit Definition to open the source location where the entity is declared.</li> </ul>
	• Listing open views: You can choose Window > Windows from the menus or look at the tabs across the top of the document area to see a list of all the separate graphical views you have open.
	<ul> <li>Scrolling: You can scroll around a graphical view by dragging your cursor within the view when you have selected Screen Drag Mode by clicking the minimum icon.</li> </ul>

• Expanding hierarchy: You can expand and contract tree views by clicking the red circle to the right of a node. Right-click on the background of a view and choose Open All Nodes or Close All Nodes to expand or contract all nodes at once. Choose Close Unselected Nodes to contract nodes that are not in the path of any selected nodes. Hold down the Ctrl key if you want to keep multiple nodes open.



• **Path highlighting:** To highlight the path for one or more entities in a tree view (such as a Callby view), select the entity and right-click. In the context menu, choose **Highlight Path**. Hold down the Ctrl key to select multiple entities for path highlighting.



- Zooming: You can zoom in or out using the toolbar.



• **Printing and saving:** Everything you see can be printed or saved. Printing may be done to one page (squeezing the picture) or across multiple pages (poster style).

See *Printing Graphical Views* on page 278 for details on printing. Graphical views can be saved as BMP, JPEG, PNG, Visio XML, and DOT files. See *Saving Graphical Views* on page 275 for details on saving to a graph file.

• Layout control: Layout is done automatically; there is no need to move lines or boxes around for a better view. Options are available for changing the layout. For example, you can control whether entities are sorted according to their order in the code or alphabetically. See *Controlling Graphical View Layout* on page 263.

-----

**Filtering Out Entities** You can apply filters to hide certain entities in some graphical views. To create such a filter, follow these steps:

1 Right-click on the background of a graphical view and choose **Edit Graphic Filters** from the context menu. (Note that this option is not available for some types of graphs. For example, it is available for Call graphs and Declaration graphs.)

🛓 G	raphic Filter Dialog	2 <b>-</b> 2	X
<b>V</b>	Enable Project Filters		
Pr	oject Filters:		_
	Filter Name	Criteria Action New	
1	copy*	Long Name Hide Node Edit	
2	Z*	Long Name Hide Node Remove	
3	gz*	Long Name Hide Node	
4	add*	Long Name Hide Node 👻 🛛 Remove All	
		OK Cancel Apply	

- 2 In the Graphic Filter dialog, put a checkmark in the Enable Project Filters box.
- 3 Click New. This opens the Graphic Filter Editor dialog.

🧸 Graphic Filt	ter Editor
Filter Text:	
Filter Criteria:	Long Name 🔹
Filter Action:	Hide Node 💌
	OK Cancel

**4** Type a filter in the **Filter Text** field. For example, use gr\* to match entity names beginning with gr. Filters are case-sensitive.
	5 In the Filter Criteria field, select whether to compare the filter to long names, definition files, or the type text of entities. For example, if you choose long names, a filter of print* does not match SomeProc::printWide. Instead, you can type *print*.
	6 In the Action field, select one of the following options:
	- Hide Node: Items that match the filter are not included in the output.
	- <b>Hide Sub Nodes:</b> The item that matches the filter is shown, but any subnodes of these items are removed from the output.
	- <b>Collapse Sub Nodes:</b> Any subnodes of items that match the filter are collapsed in the output. An icon is shown after the node to indicate that there are subnodes. Items that match the filter are shown.
	7 Click <b>OK</b> to add the filter to the project.
	You can also create filters by right-clicking on an entity in a graphical view and choosing one of the filtering options. The options allow you to quickly filter out entities with that name or in that file.
	You can remove filters you have created by clicking Remove or Remove All.
	The filters you create apply to all graphical views that support filtering. You can temporarily disable filtering in the Graphical Settings dialog or by right-clicking on any graphical view and choosing <b>Disable Graphic Filters</b> from the context menu.
Reuse Checkbox	The <b>Reuse</b> checkbox controls whether a view is reused or a new window is opened when another graphical view is requested. The Reuse box is unchecked by default. At most one graphical view can have the <b>Reuse</b> box checked at any time.
	You can cause views to be reused if a similar type of graphical view is opened from within a graphical view, no matter whether the <b>Reuse</b> box is checked. Change this behavior in the User Interface > Windows category of the <b>Tools &gt; Options</b> dialog (page 96).
Sync Checkbox	The <b>Sync</b> checkbox controls whether this graphical view changes when a different entity is selected in the Project Browser, Entity Filter, and other windows that let you select an entity. For example, if you check the Sync box in a Declaration graph window and then select a different entity in the Entity Filter, the graph shows declaration information for the newly selected entity.
Graph Options	See page 118 for information about the <b>Graphs</b> category in the <b>Tools &gt; Options</b> dialog. You can control how the display of relationships between graph nodes changes when you hover the mouse over a graph or double-click on a node.

### **Types of Views**

There are two main types of graphical views: hierarchy views and structure views.

(Dependency graphs are a separate type of graph described in the chapter on architectures. See page 196 for details. Project overview graphs are described on page 246.)

**Hierarchy View Types** *Hierarchical* views show multi-level relationships between entities. *Understand* offers hierarchy graphs of the following types of relationships. Some types apply to specific source languages.

- **Butterfly:** Shows both calls and called by.
- Calls: Shows who this entity calls.
- Calls Relationship: Show the call relationships between two entities.
- Called By: Shows who calls a given entity.
- Calledby Relationship: Show the callby relationships between two entities.
- Include: Shows who this file includes.
- IncludeBy: Shows who includes this file.
- Depends On Graph, Depended On By Graph, and Butterfly Graph: Available for classes, packages, and architectures only. See page 196 for architecture graphs.
- Derived Classes: Shows classes derived from a given class.
- Base Classes: Show what classes are the base for a class.
- Extended By: Shows which classes are extended by this class.
- Class Inheritance: Shows who inherits from a given class.
- Child Lib Units: Shows Child Library Units of a compilation unit. (Ada 95 only)
- Declared In: Shows the declaration tree from where this program unit is declared.
- Declaration Tree: Shows declaration nesting of program units in a compilation unit.
- Instantiated From: Shows instantiation tree of a generic type or compilation unit.
- Instantiations: Shows who instantiates a given generic unit.
- Invocation: Shows what compilation units a unit invokes.
- Parent Lib Unit: Shows the parent lib units of a given entity.
- Type Derived From: Shows tree of types a type is derived from.
- Type Tree: Shows types that derive new types from an entity.
- With: Shows what compilation unit an entity "Withs" into scope.
- WithBy: Shows what compilation units "Withs" a given entity.
- Uses: Shows which modules use this item.
- Used By: Shows which modules are used by this item.
- Cluster Call Internal: Shows call relationships within a file.

. . . . . . .

- Cluster Call: Shows who this entity calls.
- Cluster Callby: Shows who calls this entity.
- Cluster Call Butterfly: Shows both calls and called by.

Hierarchy ViewHierarchy views show multi-level relationships between entities. Here are examples of<br/>the types of hierarchy views that Understand offers.

• **Butterfly:** Shows both calls and called by relationships if they exist. The selected entity is outlined in red.



• **Calls:** Shows the entire chain of calls emanating from this function. Each line between entities is read as "x calls y".



• **Called By:** Shows what calls an entity. Each line connecting an entity is read as "x is called by y". In this example, error is called by code (and others), which is called by rules (and others). Note that this view is read from the bottom up or right to left.



• **Calls Relationship / Calledby Relationship:** Shows the call or callby relationships between any two entities. First, right-click on the first entity and select the graph you want to view. Then, click on another entity whose relationship to the first entity you want to find. You can click on the second entity anywhere in the *Understand* interface. The entity name will appear in the "Select a second entity" dialog. This example shows the callby relationship from the deflate() function to main().



• **Include:** Shows the include hierarchy of an entity, such as a file. A connecting line is read as "x includes y." In this example, align.h includes global.h.

🛃 Include Graph : align.h 🗙 🗹	1114	🛃 Includeby Graph : align.h 🗙
🗊 💩   🐴 🖄   🔍 🕶   🗖 Reuse 🔲 Sync	Ť	问 💩   🐴 🖄   🔍 🗸 🗌 Reuse 🗌
align.h global.h		<pre> e algebra.h e containers.h memory.cpp e align.h pl.cpp shader.cpp </pre>

- **Include By:** Shows the include tree in the other direction. In the previous example, align.h is included by several files such as algebra.h.
- **Base Classes:** For classes, shows the base classes from which this class is derived from. In this example, class *CLInearCurve* is derived from class *CCurve*, which is derived from class *CSurface* and so on.



• **Derived Classes:** Shows the classes that are derived from this class. In this example, class *CTexture3d* is a base class for classes *CIrradianceCache* and others.



• Extended By: Shows which classes are extended by other classes. A line is read as "class is extended by class." In this example, the *regexp.REToken* class is extended by a number of classes, including the *regexp.RE* class, which in turn is extended by the *regexp.UncheckedRE* class.



Structure View Types	<i>Structure</i> views offer a one glance way to see important structure and relational information about a given entity. <i>Understand</i> structure views include the following:
	• Graph Architecture: Shows the hierarchy of an architecture node. See page 199.
	• <b>Declaration:</b> Shows what a structure is composed of. For example, shows the parameters, return type, and callbys of a function. For classes, shows what members are provided, who inherits this class, and who it is based on.
	• <b>Parent Declaration:</b> Shows what a structure is composed of. Shows Calls instead of the Called Bys shown by a Declaration graph.
	• <b>Declaration File:</b> Shows what entities (such as functions, types, macros, and variables) are defined within a given file.
	Declaration Type: Shows what a type is composed of.
	Class Declaration: Shows the members defining the class and the parent class
	• Data Members: Shows what components a class, struct, or type contains.
	• <b>Control Flow:</b> Shows a flow chart of the function or similar entity type. Clicking on a node in the graphs jumps to the line of code referenced.
	• <b>Cluster Control Flow:</b> Shows a flow chart of the function or similar entity type. This view type is more interactive than the Control Flow view.
	• <b>UML Class Diagram:</b> Shows the classes defined in the project or a file and related classes. Adheres to the Unified Modeling Language (UML) structure diagram format.
	• UML Sequence Diagram: Shows interactions between entities arranged by time sequence. This graph is available for functions and methods that call member methods. See the Scientific Toolworks website for a sample UML Sequence diagrams and information about how events are displayed.
	• Package: Shows what entities are declared in a given package (body or spec).
	<ul> <li>Task: Shows the parameters, invocations, and what entities/entry points are declared in a task. Also shows what the task Withs.</li> </ul>
	Rename Declaration: Shows what entities are renamed in the entity.
Structure View	Structure views quickly show structure and relations.
Examples	Understand structure views are designed to present essential information about an entity in a small and concise manner. The structure diagram is derived from the graphs presented by Booch and Buhr in their respective books "Software Engineering with Ada" and "System Design in Ada." Where needed, symbols and annotations have been extended or altered to represent new kinds of information available from Understand.

• **Declaration:** Shows the structure of the entity. For example, shows the parameters, return type, and callbys of a function.



• **Parent Declaration:** Similar to a Declaration graph but shows what the entity calls.



• **UML Class Diagram:** Shows the classes defined in the project or a file and related classes. Right-click to show or hide class details, related classes, and solo classes.



• **Declaration File:** Shows the entities declared in the file. Also shows files included by the file and classes imported by the file.



• **Declaration Type:** Shows information about a type declaration.



• **Class Declaration:** Shows the members defining the class and the parent class from which it is derived.

Base Classes:		
(CExpression)		
l	Class: CBuiltinExpre	ession
	defined as:	
	CBuiltinExpression	CList * arguments
	~CBuiltinExpression	CFunctionPrototype * function
	void getCode	char * replacementPrototype

• **Control Flow:** Shows a flow chart of the function or similar entity type. As the following figure shows, a number of specialized options can be set when you right-click on this type of graph.



### **Graphical Notation**

The following symbols are used by *Understand* to represent various language constructs. The symbols vary somewhat depending upon the type of view.

- Entities such as functions and other program units are shown in rectangles.
- Files and system-level entities are usually shown in parallelograms.
- · Classes and types are shown in flattened hexagons.
- Macros are usually shown in flattened octagons.
- Objects such as variables are usually shown in slightly rounded rectangles.
- Unknown or unresolved entities are drawn with dashed outlines or in gray.
- Other shapes are language-specific.

In Control Flow views, standard flow chart symbols, such as diamonds for decision points, are used.

### **Controlling Graphical View Layout**

This section applies to non-cluster graphs. For information about using cluster graphs, see Controlling Cluster Graph Layout on page 272.

The two main types of graphical view windows, *Hierarchy* and *Structure*, have a variety of configuration options. You can set them by right-clicking on the background of a graphical view and choosing the option you want to modify from the context menu.



These options control the layout and drawing of the graphic views. The following subsections describe a number of these options. The list of available options varies depending on the type of view.

Note that the options for Cluster graphs are different, and are described in Controlling Cluster Graph Layout on page 272.

. . . . . . . . . . . . .

Called by Menu	The <b>Called by</b> menu controls whether program units that call the current entity are shown in declaration views.			
	View with Called By set to On	View with Called By set to Off		
Q	Called By: GetoptDemo.main	LongOpt.getName returns: String		
Comments Menu	The <b>Comments</b> menu controls whether to show a entity. The default is Off.	any comments associated with an		
Constants Menu	The <b>Constants</b> menu controls whether to show controls whether to show controls of the show control of the	onstants in Declaration views. The		
Default Members Menu	The <b>Default Members</b> menu controls whether de members of the class.	claration views show default		
Dependent Of Menu	The <b>Dependent Of</b> menu controls whether files a the C File Declaration view. The default is On.	C file is dependent on are drawn in		
Dependent Menu	If <b>Dependents</b> is on (the default) then files dependente a File Declaration view.	dent on the current C file are shown in		
Depth	Sets the number of levels to which a dependency level.	graph is expanded. The default is 1		
Duplicate Subtrees Menu	The <b>Duplicate Subtrees</b> menu controls whether in tree are shown in hierarchy views. The options are default is to show duplicate subtrees. In some app can dramatically simplify hierarchy views. Duplicate has over 1000 nodes.	multiple occurrences of the same sub- e to Hide or Show such subtrees. The plications, hiding duplicate subtrees te subtrees are not shown if a view		

Expand Recursive Notes	Controls whether recursive nodes in a dependency graph are shown as separate items. The default is to show the expansion for recursive nodes. If you turn this setting off, a particular item is expanded only at the highest level where it occurs in the architecture, class, or package hierarchy. Unexpanded nodes that are recursive at lower levels display "(recursive)" as part of their text.
Expand Repeated Notes	Controls whether repeated nodes in a dependency graph are shown as separate items. The default is to show the expansion for repeated nodes. If you turn this setting off, a particular item is expanded only at the highest level where it occurs in the architecture, class, or package hierarchy. Unexpanded nodes that are repeated at lower levels display "(repeated)" as part of their text.
Extended By Menu	The <b>Extended By</b> menu controls whether declaration views show classes by which the selected class is extended.
Extends Menu	The <b>Extends</b> menu controls whether declaration views show classes that the selected class extends.
External Functions Menu	If <b>External Functions</b> is on then functions defined in a header file or in a file included by a header file are shown in the Declaration View for a header file. The default is On.
Filename Menu	The <b>Filename</b> menu controls how filenames are displayed in views. It is available for both declaration and hierarchy views.
	None: Filenames are not shown in the view.
	<ul> <li>Shortname: Where filenames are relevant, only the name of the file is shown in square brackets.</li> </ul>
	• Fullname: Where filenames are relevant, the full file path and filename are shown in square brackets.
Function Pointer Menu	The <b>Function Pointer</b> menu controls whether function pointers are displayed as invocations in the Call and CallBy trees.
Globals Menu	The <b>Globals</b> menu controls whether to show globals in Declaration views. The default is On.
Implements Menu	The <b>Implements</b> menu controls whether declaration views show entities that the selected entity implements.
Implemented By Menu	The <b>Implemented By</b> menu controls whether declaration views show entities by which the selected entity is implemented.

Imports Menu The **Imports** menu controls whether declaration views show entities imported by the current entity. View with Imports set to On View with Imports set to Off Imports: GetoptDemo.java getopt.LongOpt GetoptDemo getopt.Getopt GetoptDemo.java GetoptDemo **Included By Menu** If **IncludeBy** is on (default) then files that include the Header File being drawn in a Header File Declaration view are shown. Includes Menu The Includes menu controls if include files are drawn on file declaration diagrams (C file, Header file). The default is On. Inherits Menu The Inherits menu controls whether declaration views show entities that the selected entity inherits. **Inherited By Menu** The Inherited By menu controls whether declaration views show entities inherited by the selected entity. The Intrinsic menu controls whether intrinsic functions (for example, cos and sin) are **Intrinsic Functions** Menu displayed or hidden. Invocations Menu The **Invocations** menu controls whether procedures and functions called by the current procedure or function are shown in Declaration views. **View shows Invocations** View without Invocations shown Subroutine arange Subroutine arange Called By: Called By: genera cos genera (Unnamed Main) (Unnamed Main) sin float

sqrt

The **Layout** menu controls the layout algorithm for a hierarchical chart. It is available only in hierarchy views (calls, callby, etc.). The options are:

• **Crossing:** A left-to-right view, minimizing space used but sacrificing some readability by permitting lines between entities to cross.



• **Horizontal Non-Crossing:** A left-to-right layout, using more space in some situations but enhancing readability by having no crossing lines.



• Vertical Non-Crossing: A top-to-bottom layout similar to Horizontal Non-Crossing.



Layout Menu

#### Level Menu

The **Level** menu controls the number of levels to be traversed when laying out a hierarchical view. The default value is "All Levels". Values of 1 to 5 may be set. It is available only in hierarchy views.





Sort Menu	The <b>Sort</b> menu lets you specify whether enables alphabetically. If this option is off (the default encountered in the project.	ntity names in tree views should be sorted ult), entities are sorted in the order they are	
Spacing Menu	The <b>Spacing</b> menu lets you choose to cha choose compact, small, normal, wide, or e	inge the space between boxes. You can xtra wide.	
Sql Menu	The <b>SqI</b> menu lets you specify whether SQL entities should be shown in graphical views. This option is on by default.		
Static Menu	The <b>Static</b> menu controls if static functions File declaration views. Static functions are They are visible only within the file they are drawn with the edge of their box inside the enclosing unit (C file). The default is On.	s are drawn in function, C File and Header those declared using the "static" keyword. e declared in. If enabled static functions are edge of the outer declaration box for their	
Text Menu	<ul> <li>The Text menu sets the way entity names are trimmed or altered to accommodate the layout of graphics. It is available for both declaration and hierarchy views. Names may be truncated to a certain length or wrapped at a certain length.</li> <li>No Truncation: Uses the name as defined in the source code. This is</li> </ul>	Called By  Calls Calls Filename Function Pointer Name Spacing No Truncate	
	<ul> <li>the default.</li> <li>Truncate Short: Cuts off names at 10 characters.</li> </ul>	Unresolved Edit Graphic Filters	
	• <b>Truncate Medium:</b> Cuts off names at 20 characters.	Truncate Long No Wrap	
	• <b>Truncate Long:</b> Cuts off names at 30 characters.	Wrap Short Wrap Medium	
	No Wrap: Never wraps text to the next line.	Vrap Long	
	<ul> <li>Wrap Short: Wraps the name between 8 and 10 characters. Location in depends on if a natural wrapping character is found. Natural wrapping c are and :</li> </ul>		
	ept wrapping range is 15-20 characters.		
Types Menu	The <b>Types</b> menu controls whether to show default is On.	v types in Program Declaration views. The	

Typetext Menu	The <b>Typetext</b> menu tells declaration views (Function Declaration, C File Declaration, Header File Declaration) to include types on the view. The default is On.
Unknown Menu	The <b>Unknown</b> menu controls whether entities should be shown if they are used in the source, but are never declared or defined.
Unresolved Menu	The <b>Unresolved</b> menu controls whether entities should be shown if they have been declared but not defined. For example, an entity may be declared in a header file, but never defined in the source. This option is available on hierarchy and structure views.
	Unresolved include files are those that are included but not found along a declared include path (either a compiler or project include path).

Unresolved entities are drawn as normal but with a dashed border:



Usedby Menu	The <b>Usedby</b> menu tells Declaration views whether to show items that use this item.
Uses Menu	The <b>Uses</b> menu tells Uses views whether to show only items that are used directly, or to also show items that are used by nested subprograms. The default is to show both.
Variables Menu	The <b>Variables</b> menu controls whether to show globals in Declaration views. The default is On.
Withs Menu	The <b>Withs</b> menu controls on Declaration views of compilation units (packages, tasks, separate procedures, etc) if Withs are drawn. The default is On.
With Bys Menu	Controls if <b>With Bys</b> (who Withs a given compilation unit) are shown on Declaration views. The default is On.

### **Controlling Cluster Graph Layout**

Cluster graphs are a special type of hierarchy view. They provide a more interactive view of call relationships than other hierarchy views. The **Cluster Call, Cluster Callby**, **Cluster Call Butterfly, Cluster Call Internal**, and **Cluster Control Flow** variants are available, and can be accessed from the function, class, file, or architecture level. The Architecture Dependency graphs (page 196) behave similarly to the cluster graphs. There are some special menus for Cluster Control Flow graphs that are described on page 274.

For example, if you open a Cluster Call Butterfly graph for a file, you see a graph similar to the following:



If you then double-click on some of the file boxes, you can see call relationships for functions within the files that you expand.



Hold down Ctrl+Shift and use the mouse wheel to zoom in and out in cluster graphs.

Hold down Alt and click on an entity to go to the code that defines that entity.

The toolbar for cluster graphs is the same as for other graphs, and the context menu for entities in the graph is similar to elsewhere.

The context menu when you right-click on the background of a cluster graph offers the following options:

- Aggregate Nodes by: Choose an architecture you want to organize entity nodes.
- Edges Shown: Choose which relationships to the originally selected entity you want shown. "Forward" is call relationships. "Reverse" is callby relationships. "Butterfly" is both call and callby relationships.
- Entity Name Format as: Choose whether you want to display short or long names for entities.
- **Highlight Paths to Selected Node(s):** You can highlight all paths between the node for which you opened a Cluster Call or Cluster Callby graph and some other node. To do this, select a node (not the original node), right-click on the background of the graph (not on an entity or within a box), and choose this option. You can hold down the Ctrl key to select multiple entities for path highlighting



- Include Virtual Edges: Set this item to On if you want to show override and overriddenby edges. Such edges are light blue by default. If you want to change this color, choose Tools > Options. In the Graphs category, make sure Use custom style on cluster call graphs is checked. Then.create a new Edges style called "overrides,overriddenby".
- Show Edge Labels: Set this item to On if you want the number of occurrences of this relation to be shown in the graph. For bi-directional call relationships, the two numbers in the label show calls in each direction.
- Show Edges Between Children By Default: Set this item to On if you want to show inter-child edges initially for nodes that are expanded. Note that changing this setting only affects nodes that are using the defaults; You may need to click the Restore Defaults icon to affect the entire graph.
- **Show Legend:** Set this item to On if you want to show a graph legend in the upper left. The legend identifies the shapes and arrow styles used in the graph.
- Show Node Children By Default: Set this item to On if you want nodes to be opened by default when you open a cluster graph. For example, all functions within files will be shown by default if this option is enabled when you open the Cluster Callby graph for a file.

The context menu when you right-click on a node in a cluster graph offers commands similar to those you see when right-clicking on an entity name elsewhere in *Understand*.

The context menu when you right-click on an edge (arrow) in a cluster graph provides a list of the references that constitute the edge. Choose an item from the list to visit the source code for this relationship. You can limit the length of this list as described for the **Tools > Options** dialog on page 118.

The Graph Customizer to the right of a cluster graph offers the same settings as those described for Architecture Dependency Graphs in *Graph Customizer Toolbar* on page 197.

You can customize the display colors, shapes, and arrows used in cluster graphs in the Graphs category of the **Tools > Options** dialog (page 118).

**Cluster Control Flow** Cluster Control Flow graphs start Graphs show the execution flow of Cluster reginsert an entity such as a function. ✓ Collapse These graphs have the  $\checkmark$ Show Comments same toolbar as other Debug graphs and a Graph register char \*src register char \*place Customizer toolbar  $\checkmark$ Show Entity Name register char \*dst (page 197) similar to those Allow Call Expansion in other Cluster Graphs and √ Filter Architecture Dependency Graphs. However, the ✓ Show Labels regcode == &regdummy context menu when you Expand Macros right-click includes the following options: ✓ Show Source Code ves no Styled Labels • Allow Call Expansion: Allows called functions to src = regcode be expanded by clicking. \*--dst = \*--src dst = regcode regsize += regcode += 3 If this option is on, expandable calls are shown as a 3D box. Off yes by default. Cluster: Uses a box to src > opnd return enclose statements in a group such as the "if" or "else" branch of a conditional statement. On by default. Collapse: Combines statements into a single box if there are no decision points between them. On by default. Debug: Shows details about the information about each item in the flow. In order, the detail information is: nodeID, nodeKind, startLine, startCol, endLine, endCol, endNode, commaSepararatedListOfChildren. Off by default.

- Expand Macros: Enabling this option shows macros expanded if you have enabled the Save macro expansion text option in the C++ project configuration (page 68). Off by default.
- Filter: Hides implicit actions, such as "endif". On by default.
- Layout: Choose whether to arrange the graph vertically or horizontally. The default is Vertical.
- Show Comments: Shows comments associated with statements in the graph. On by default.
- Show Finally-Block Flows: Shows edges representing exceptional exits from a trycatch block in languages like Java and C#. On by default.
- Show Entity Name: Shows the name of the entity in the Start box at the beginning. Off by default. You can also choose to show entity names with parameters included.
- Show Labels: Shows text for edges (for example, yes/no) and start block. On by default.
- **Show Legend:** Set this item to On if you want to show a graph legend in the upper left. The legend identifies the shapes and arrow styles used in the graph.
- Show Source Code: Shows source code in boxes. On by default.
- **Styled Labels:** Highlights keywords, comments, and strings in source code shown in the graph. The formats defined in the Editor > Styles category of the Understand Options dialog (page 115) are used. On by default.

#### **Saving Graphical Views**

*Understand* offers a number of ways to export your graphical views and use them in other ways. The toolbar for each graphical view provides the following icons for copying and printing graphs.



In addition to printing, you can save graphical views as JPEG, PNG, SVG files (page 276), Visio XML files (page 276), and DOT files (page 277). The first three formats are common graphics formats.

For cluster graphs, you can save and load customized graph settings as described for Dependency graphs (page 197).

<ul> <li>Saving Views to Files</li> <li>To save a graphical view in one of the following formats, use the Export drop-down the graphical view toolbar to choose the Export to Image File option. Or choose File &gt; Export to Image File form the menus. In the Export dialog, choose a location, filename, and file type for the view.</li> <li>Image File from the menus. In the Export dialog, choose a location, filename, and file type for the view.</li> <li>JPEG files are compressed bitmaps. They can be viewed with most web browsers, document editors, and graphics programs. This format is "lossy"; some data is lost in the compression.</li> <li>PNG files store compressed bitmaps similar to GIF files. They can be viewed with most web browsers, document editors, and graphics programs. They use a non-patented compression method.</li> <li>SVG files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.</li> <li>You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit &gt; Copy Image to Clipboard from the menus. Then, paste the image into another program.</li> <li>Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.</li> <li>Saving Views as Visio Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.</li> <li>You do not need to have Visio installed in order to save a graphical view toolbar to choose the Export drop-down the graphical view toolbar to choose the Export drop-down the graphical view toolbar or choose elote a smaller size.</li> </ul>				
<ul> <li>Svort To Image File</li> <li>LEXport To Visio XML Export To Visio XML Export to .Dot</li> <li>JPEG files are compressed bitmaps. They can be viewed with most web browsers, document editors, and graphics programs. This format is "lossy"; some data is lost in the compression.</li> <li>PNG files store compressed bitmaps similar to GIF files. They can be viewed with most web browsers, document editors, and graphics programs. They use a non- patented compression method.</li> <li>SVG files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.</li> <li>You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit &gt; Copy Image to Clipboard from the menus. Then, paste the image into another program.</li> <li>Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.</li> <li>Saving Views as Visio</li> <li>Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.</li> <li>You do not need to have Visio installed in order to save a graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.</li> </ul>	Saving Views to Files	To save a graphical view in one of the following formats, use the <b>Export</b> drop-down the graphical view toolbar to choose the <b>Export to Image File</b> option. Or choose <b>File &gt; Export to Image File</b> from the menus. In the Export dialog, choose a location, filename, and file type for the view.		
<ul> <li>JPEG files are compressed bitmaps. They can be viewed with most web browsers, document editors, and graphics programs. This format is "lossy"; some data is lost in the compression.</li> <li>PNG files store compressed bitmaps similar to GIF files. They can be viewed with most web browsers, document editors, and graphics programs. They use a non-patented compression method.</li> <li>SVG files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.</li> <li>You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit &gt; Copy Image to Clipboard from the menus. Then, paste the image into another program.</li> <li>Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.</li> <li>Saving Views as Visio</li> <li>Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.</li> <li>You do not need to have Visio installed in order to save a graphical view as a Visio XML file.</li> <li>To save a Visio XML file, use the Export drop-down the graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.</li> </ul>		Export  Export To Image File Export To Visio XML Export to .Dot		
<ul> <li>PNG files store compressed bitmaps similar to GIF files. They can be viewed with most web browsers, document editors, and graphics programs. They use a non-patented compression method.</li> <li>SVG files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.</li> <li>You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit &gt; Copy Image to Clipboard from the menus. Then, paste the image into another program.</li> <li>Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.</li> <li>Saving Views as Visio</li> <li>Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.</li> <li>You do not need to have Visio installed in order to save a graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML file is *.vdx.</li> </ul>		• <b>JPEG</b> files are compressed bitmaps. They can be viewed with most web browsers, document editors, and graphics programs. This format is "lossy"; some data is lost in the compression.		
<ul> <li>SVG files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.</li> <li>You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit &gt; Copy Image to Clipboard from the menus. Then, paste the image into another program.</li> <li>Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.</li> <li>Saving Views as Visio</li> <li>Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.</li> <li>You do not need to have Visio installed in order to save a graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.</li> </ul>		<ul> <li>PNG files store compressed bitmaps similar to GIF files. They can be viewed with most web browsers, document editors, and graphics programs. They use a non- patented compression method.</li> </ul>		
You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the Copy icon on the graphical view toolbar or choose Edit > Copy Image to Clipboard from the menus. Then, paste the image into another program. Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.Saving Views as Visio FilesMicrosoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications. You do not need to have Visio installed in order to save a graphical view as a Visio XML file. To save a Visio XML file, use the Export drop-down the graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.		• <b>SVG</b> files are Scalable Vector Graphics files. This file type uses XML to describe a 2-dimensional vector-based image.		
Saving Views as VisioMicrosoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications. You do not need to have Visio installed in order to save a graphical view as a Visio XML file.To save a Visio XML file, use the Export drop-down the graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.		You can also copy a graphical view to the clipboard and paste it as a bitmap into the image program or word processor of your choice. To do this, click the <b>D</b> Copy icon on the graphical view toolbar or choose Edit > Copy Image to Clipboard from the menus. Then, paste the image into another program.		
Saving Views as VisioMicrosoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications. You do not need to have Visio installed in order to save a graphical view as a Visio XML file. To save a Visio XML file, use the Export drop-down the graphical view toolbar to choose the Export to Image File option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.		Note that if the graph would result in an image larger than 200 MB, the graph will be resized to a smaller size.		
You do not need to have Visio installed in order to save a graphical view as a Visio XML file. To save a Visio XML file, use the <b>Export</b> drop-down the graphical view toolbar to choose the <b>Export to Image File</b> option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.	Saving Views as Visio Files	Microsoft Visio is a vector-based graphics program used for drawing flowcharts and similar graphics. That is, it deals with shapes and objects rather than pixels. Visio XML is an Extended Markup Language that is supported by Visio and a number of other graphics applications.		
To save a Visio XML file, use the <b>Export</b> drop-down the graphical view toolbar to choose the <b>Export to Image File</b> option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.		You do not need to have Visio installed in order to save a graphical view as a Visio XMI file.		
		To save a Visio XML file, use the <b>Export</b> drop-down the graphical view toolbar to choose the <b>Export to Image File</b> option. In the Export dialog, choose a location and filename for the view. The file extension for Visio XML files is *.vdx.		
Funant -		Funant a		
Expert To Image File		Expert To Image File		
Export To Visio XML		Export To Visio XML		
Export to .Dot		Export to .Dot		

 Saving Views as DOT
 DOT is a language used to describe graphs in plain text. This format can be imported and edited by a number of external tools. You can export many (but not all) types of graphs produced by Understand to a DOT file.

To save a DOT file, use the **Export** drop-down the graphical view toolbar to choose the **Export to .Dot** option. In the Export dialog, choose a location and filename for the view. The file extension is \*.dot.



If this option is not shown in the **Export** drop-down, the current graph cannot be exported to the DOT format.

### **Printing Graphical Views**

Understand has these printing modes:

- **Source File** printing sends a text file to the printer using 66 lines of source per page. See *Printing Source Views* on page 190.
- **Graphical view printing** provides options for how to fit the image to a page. See *Graphical View Printing* on page 278.

Graphical ViewTo print the current graphical view, you can click Print icon on the graphical viewPrintingtoolbar. Or, choose File > Print Entity Graph from the menus.

When you choose to print a graphical view, you see the Graphic Print Options dialog.

🔏 Graphic Print Options		
Printing options		
Full size (1 x 2 : 2 pages total)		
Fit to single page (1page total)		
Scale by: 100 - % (1 x 2 : 2 pages total)		
Save to pdf file : rCallButterflyGraph-crc32-c.pdf		
☑ Print page number identifiers		
Print page numbers in margin area		
Print page border markers		
Printer Settings		
Printer: Epson Stylus NX510(Network)		
Settings: Portrait, Letter (8.5 x 11 inches, 216 x 279 mm)		
Printer Page Setup		
OK Cancel		

You can choose to print the image at one of the following sizes:

- **Full size** uses the default scaling of 100%. The dialog shows the number of pages in width x height format. The page size selected with Page Setup is used.
- Fit to a single page scales the image to fit on the selected page size.
- Scale by lets you choose the sizing percentage and shows the number of pages that will be printed.

Check the **Save to PDF** file box if you want the image saved to an Adobe Acrobat file rather than being sent to a printer. This PDF printing feature does not require that you have third-party PDF generating software installed on your computer.

Check the **Print page number identifiers** box if you want page numbers on each page in the upper-left and lower-right corners. The page numbers are in "(column, row)" format. For example, (1,3) indicates that the page goes in the leftmost (first) column of the third row when you piece the pages together. The page number is not printed if the view fits on a single page.



Check the **Print page numbers in margin area** to place the page numbers outside the borders of the graph. If this box is unchecked, page number indicators are printed just inside the border markers.

Check the **Print page border markers** box to place corner markers in each corner of each page.

Click the **Printer** button to open the standard Print dialog for your operating system. When you click **Print** or **OK** in that dialog, you return to the Graphic Print Options dialog.

Click the **Page Setup** button to open a Page Setup dialog, which allows you to choose the paper size, paper source (if applicable), page orientation, and margin width. Click **OK** to return to the Graphic Print Options dialog.

Click the **OK** button in the Graphic Print Options dialog to send the graphical view to the printer (or a PDF file).

*Note:* The **File > Page Setup** menu option applies only to printing source code and other text files. The **Page Setup** button on the Graphic Print Options dialog saves its settings separately.

## Chapter 11 Using CodeCheck for Standards Verification

This chapter is explains how to use CodeCheck to find places where your code does not conform standards you select.

This chapter contains the following sections:

Section	Page
About CodeCheck	281
Running a CodeCheck	282
Viewing CodeCheck Results	285
Using CodeCheck Configurations	292
Writing CodeCheck Scripts	293

### About CodeCheck

*Understand* provides a tool called CodeCheck to make sure your code conforms to published coding standards or your own custom standards. These checks can be used to verify naming guidelines, metric requirements, published best practices, or any other rules or conventions that are important for your team.

Ο ι	Inders	stan	nd C	odecheckx							►UT
Resul	ts by F	File:		Show Violation Counts	Sho	ow Ignore	ed Viola	ations 🔽	Flatten Files List	è 🔨	θ
				Results		Entity	Line	Column	Check		*
31	$\triangleright$	)	C;\s	sample\fastgrep\regexp.h							_
3	4		C;\s	sample\fastgrep\regmagic.h							
		$\triangleright$	Θ	Octal Constants not allowed	:	MAGIC	5	16	Do not use Octal con	stants	=
		$\triangleright$	Θ	#define used		MAGIC	5	0	Do not use #define to	o speci	
		Þ	Θ	Missing wrapper in header f	ile.	regm			Ifndef Wrappers		
52	$\triangleright$		C:\s	sample\fastgrep\regsub.c							-
The s	The string FILENAME_H or FILENAME_HPP is in the symbol following the first #ifndef in all header files.										
Reco	Reconfigure         Result Log         Results by Check         Result Locator         Result Treemap										

Checks are available to make sure your code conforms to several published coding standards. You can select a subset of individual checks to test for from these standards. For example, you can check to make sure that all if...elseif constructs contain a final else clause.

For all languages, checks are provided to let you verify that various entity types conform to your naming conventions and to confirm that your code meets metric requirements you set for complexity, function length, and nesting depth.

If you want to perform custom checks, you can create your own checks using Perl. For example, you can create a check to find lines longer than 80 characters or filenames that begin with a number.

CodeCheck validation suites are available from the SciTools website.

### **Running a CodeCheck**

To open the CodeCheck tool, choose **CodeCheck > Open CodeCheck** from the menus.

**Files Tab** 

In the **Files** tab, choose whether to check all files in the project, only files that have changed since you last ran CodeCheck, only files that have changed since a specific date, or the files you select.

Understand C	odecheckx	FIII
	Select files to analyze:	
	🔿 All Files 🙌 💼 మ	
Files	Files that have changed since the last run: Monday, March 19, 2012 10:57:30 AM	
	◎ Files that have changed since: 3/7/2012 6:13:17 PM	
Checks	I choose which files to use	
Ignores List	▲       ✓       fastgrep         ✓       egrep.c         ✓       regexp.c         ✓       regexp.h         ✓       regsub.c         ✓       strpbrk.c         ✓       timer.c         ✓       try.c	
Inspect	Next	]

If no files have been changed since a date you select or since the last time you ran CodeCheck, you will see a message that says no files meet the criteria.

You can click the **Search** icon to open a search bar below the list of files that lets you search by filename.

If your project contains many files, you can more easily select specific files by following these steps:

- Export the entire list of currently selected files (with full file paths) to a text file by clicking the interval in the selected Files icon.
- 2 Edit the file with a text editor. Delete the lines for files you do not want to inspect, and save the file as plain text.
- 3 Click the **1** Import File List icon and import the file you edited. This will select only the files listed in the file.

Once you have finished selecting the files to inspect, click **Next** to move to the Checks tab.

## Checks Tab In the Checks tab, the default configuration is New and all the SciTools' Recommended Checks that apply to your programming languages are selected.

If you want to use a different CodeCheck configuration or save the configuration you are creating for later use, see *Using CodeCheck Configurations* on page 292.

Co	onfiguration: recommended 🔹 📘 🗙	· 🔥 🔁 🖻
	Check Name	Description
	Image: SciTools' Recommended Checks	Universal standards agreed upon by most exper
	a 📄 🚺 Published Standards	Published standards from recognized experts in
	Effective C++ (3rd Edition) Scott Mey	From the book "Effective C++ Third Edition" by S
	Image: Misra C_2012	MISRAC:2012 - Guidelines for the use of the C
	Image: Misra-C 2004	MISRAC:2004 - Guidelines for the use of the C
	Image: Misra-C++ 2008	$MISRA\ C\text{++}$ : 2008 - Guidelines for the use of the
	D Clang Static Analyzer (beta)	Checks provided by the clang static analyzer.
	🔺 🥅 🚺 All Checks	
	Image Specific	
	Image:	
	🔲 💿 No Control Code Characters	Report control code characters that are used in
	No direct or indirect recursion allowed.	Functions shall not call themselves, either direct

Choose the checks you want to perform. The following types of options are provided:

- SciTools' Recommended Checks: This category lists recommended checks for your source code languages. These are standards violations that we feel are most serious.
- **Published Standards:** Collections of checks are provided to see if your code conforms to the following standards or recommendations:
  - Effective C++ (3rd Edition) Scott Meyers
  - MISRA-C 2012
  - MISRA-C 2004
  - MISRA-C++ 2008
- **Custom Checks:** Any custom checks you have installed are listed in the Checks tab. See *Writing CodeCheck Scripts* on page 293.
- **Clang Static Analyzer:** Checks provided by the Clang Static Analyzer tool. These checks are most accurate when used with the strict C parser.
- Language Specific Checks: All checks that Understand can perform are listed here. They are organized first by programming language, then by category, and finally by check. Currently, most checks apply to C/C++ code, but some checks are available for other languages. You can confirm that your naming conventions are met for various entity types in most supported languages. You should check specific languages under the All Checks node; if you check the All Checks box, some of the checks may conflict with others, and errors are likely to occur.

- Metrics Checks: You can perform checks based on the values of complexity, function length, and nesting depth metrics, which are described in the list at scitools.com/support/metrics\_list/.
- **General Checks:** General checks for preferences such as lines longer than a maximum number of characters or the presence of various control characters.

Press Ctrl+F or click the **Search M** icon to be able to search for a check.

When you select a check, information about that check is shown in the **Detailed Description** area. You can copy the text in the description if you want to paste it into a report or email message. You can hide the description by clicking the arrowhead above the description.

For a number of checks, options are shown below the description. For example, if you select a metrics check, you can set a value that needs to be met. If you select a Naming Conventions check, you can specify a minimum and maximum length for acceptable names, any required prefix or suffix, and the types of characters and capitalization rules that names need to follow. The description provides details about the options.

-					
n		÷	0		
v	υ	u	υ	 3	١.

Test All Const name	8	
Minimum Length:	1	
Maximum Length:	0	
Required Prefix:		
Required Suffix:		
Character Set	II Characters	
Capitalization: ig	nore 🔹	
Consecutive Capi	tals Allowed -	

Check the **Use Verbose Logging** box if you want the Result Log to include a separate line for each violation found. Otherwise, the Result Log will present a summary and you can use the other tabs to sort through and view specific violations.

Once you have finished selecting the checks to perform, click Inspect.

You can choose **CodeCheck > Analyze Changes and Re-Run Previous Checks** from the menu bar to run the **Analyze Changed Files** command and then the **Re-Run Previous Checks** command.

If you have made changes to the checks being performed, you are asked if you want to save the configuration before performing the analysis. Click **Yes** if you might want to perform this same set of checks in the future. If you were using the "New" configuration, you are then prompted to type a name for this configuration. Type a name and click **OK**. See *Using CodeCheck Configurations* on page 292 for more information.

If more than 300,000 violations are detected, you are asked if you want to continue the check.

.....

Exporting and Importing Configurations If you want to edit the list of checks being performed using a text file, do the following:

- Export the entire list of currently selected checks (with full file paths) to a \*.ini file by clicking the Export Configuration to File icon in the Checks tab.
- 2 Edit the file with a text editor. Delete the lines for checks you do not want to apply, and save the file as plain text.
- 3 Click the **1** Import Configuration from File icon in the Checks tab and import the \*.ini file you edited. This will select only the checks listed in the file.

Exporting the configuration to a file is also required if you want to be able to perform a CodeCheck analysis from the command line.

You can export the list of checks performed to a text file by clicking the **Export** Selected Checks to Text File icon. This file includes a description of each test.

### Viewing CodeCheck Results

After you perform a CodeCheck analysis, you can view the results in the Results Log (page 285), Results by File tab (page 286), Results by Check tab (page 287), Result Locator tab (page 288), and Result Treemap (page 289). You can also print or export the results (page 290).

You can hide results by ignoring particular checks and violations (page 290).

.....

**Using the Result Log** When you run a CodeCheck analysis, you automatically see the **Result Log** tab, which provides a summary of the results.

The Result Log includes the number of files checked, how many checks were performed, and the number of violations found.

🧕 Getting Started <sub>×</sub>	🛛 🚱 try.cx 🤣 Understand Codecheck*x		FILL
	Result Log:	1	<b>F</b>
Files	Begin Analysis: Monday, March 19, 2012 3:25:05 PM File: egrep.c: Violations found File: regerror.c: Violations found File: regexp h: Violations found		
Checks	File: regsub.c: Violations found File: strpbrk.c: Violations found File: timer.c: Violations found		
Result Log	File: try.c: Violations found End Analysis: Monday, March 19, 2012 3:25:07 PM		
Results by File	Analysis Summary: Files: 8 Checks: 15		
Results by Check	Violations Found: 257 Violations Ignored: 0 Violations Remaining: 257		
Result Locator	Reconfigure Results by File Results by Check Result Locator Res	sult Tre	emap

**File Tab** 

If you checked the Use Verbose Logging box in the Checks tab, the Result Log also includes a separate line for each violation found.

You can copy the log to your clipboard for pasting by clicking the **[1] Copy** icon. To save the log to a file, click the 📄 **Export** icon.

..... Using the Results by Choose the **Results by File** tab to list the problems in each file of your project.

> The table lists the number of violations in each file and full file paths. Uncheck the Show Violation Counts box above the table to hide the number of violations. Uncheck the Flatten Files List box to organize files in a folder hierarchy that you can expand as needed.

Click the arrow next to a filename to expand the list of violations found in that file. The line for a violation shows the problem, the name of the entity, the line number on which the problem occurs, the number of the column (0 indexed) where the problem began, and a short description of the check performed.

Results by File: Expand All Co	Ilapse All 📝 Show Violation 🛛	V Show Igno	red 🗸	Flatten	Files List  🍓 🗐 😝
1	Results	Entity	Line	Column	Check
4					
1 ▷ C:\Program Files\Scī	Tools\sample\fastgrep\egrep.c				
1 b C:\Program Files\Sci	Tools\sample\fastgrep\regerror.	.c			
2 🔺 📄 C:\Program Files\Scī	Tools\sample\fastgrep\timer.c				
Violation: regerr	or defined but not called.	regerror	89	0	Unused Functions
🖉 🕘 Unused Local V	ariable	dummy	54	13	Unused Local Vari
52	int ncomp, nexec,	nsub;			
53	struct try one;				
54	<pre>char dummy[512];</pre>				
55					
56	if (argc < 4) {				

Click the arrow next to a violation to see the 5 lines of code surrounding the problem. You can double-click on the code to open the source file.

When you select a violation, the description of that check and any exceptions to the check are shown below the table. You can select text in this area and press Ctrl+C to copy it to your clipboard for pasting into other applications.

Click Expand All or Collapse All to show or hide all violation details.

Click the list or the list or the **Export** icon to send the full list of violations to HTML files in a directory you select, your clipboard, or a text file.

Check the Show Ignored Violations box if you want to see the full list of violations by overriding any ignored checks and violations. See page 290 for details.

#### Using the Results by Check Tab

The **Results by Check** tab is similar to the Results by File tab (page 286). However, all violations of a particular type are listed together. The organization under each violation is either a list of files if the **Flatten Files List** box is checked or a folder hierarchy if the **Flatten Files List** box is unchecked.

Results by Check: Expand All Collapse All 🗹 Show Violation 🔽 Show Ignored 🔽 Flatten Files List 🎂 👘 \Theta

	Results
4	A Number of Results: 4
4	SciTools' Recommended Checks
1	A Overly Complex Functions
1	4 📁 fastgrep
1	a grep.c
	<ul> <li>Program overly complex (Complexity:30)</li> </ul>
	139 char **args;
	140
	141 main(argc, argv)
	<pre>142 int argc;</pre>
	143 char *argv[];
2	Vnused Functions
1	Vnused Local Variables
•	III

If many violations of a particular type are detected, you might want to look at the individual checks in the **Checks** tab to see if you can set options to control the sensitivity of the checks. For example, for the "Magic Numbers" check, you can specify that bitfields can be set to fixed values and you call allow exceptions for values like 0 and 1. Another example is that for the "Functions Too Long" check, you can set the length that is considered too long and choose to ignore comment lines and blank lines.

In the Results by Check tab, the is **Export** icon allows you to either a detailed or summary report to HTML files in a directory you select, your clipboard, or a text file. The summary report is similar to the following:

```
CodeCheck Summary
Number of Results: 704
SciTools' Recommended Checks704
""Commented Out"" Code7
Definitions in Header Files28
Functions Too Long5
Goto Statements28
Magic Numbers300
Overly Complex Functions10
Unused Functions36
Unused Local Variables50
Variables should be commented15
```

# Using the ResultThe Result Locator tab lets you search for violations using pattern matching and<br/>sorting on the file, violation name, line number, and column number.

Result Locator:				Show Ignored Violat	tions 🍓 👘	Θ
File	<b>≜</b> ∦	Violation	▼LLL	Line 🛔	Column	
reg	×	value	×	2 -	2 -	
regexp.h		Fixed Value(1)	used inco	15	21	
regsub.c		Fixed Value(0)	used inco	73	35	
regsub.c		Fixed Value(0)	used inco	60	29	
regsub.c		Fixed Value(1)	used inco	64	30	
regsub.c		Fixed Value(0)	used inco	65	25	
•						•
Rationale Using fixed values number in choosin should be replaced	(Magic Number g that number a d with name co	) in code makes it ( ind makes it hard to nstants to make the	difficult to kno o adapt and e em more usat	w why the enginee xtend the program. ble.	er chose that Magic numbers	
13	char *reg	must;	/* I	nternal use	only. */	1
14	int regml	en;	/* I	nternal use	only. */	-
15	char prog	ram[1];	/* U	nwarranted c	humminess v	1
<pre>16 } regexp</pre>	;					-

You can type values to match filenames and violations. Right-click a column header or click the small drop-down icon to see the context menu for that column. You can choose for the filter to be case sensitive or not. You can also choose for the filter pattern matching syntax to use fixed strings (the default), wildcards, or regular expressions. To search for field values that do not contain a particular string, type ! (exclamation mark) and then the filter.

For details about using the locator fields, see Filtering the List on page 157.

Five lines of code surrounding the violation are shown at the bottom of the window. You can double-click this code to open the file.
# Using the Result Treemaps show metrics graphically by varying the size of node blocks and the color gradient. Each node block represents a code file. Different metrics can be tied to size and color to help you visualize aspects of the code. CodeCheck lets you create treemaps that show the total number or density of check

violations and the number of types of violations. For example, in this treemap larger block sizes indicate more violations in that file and darker blue indicates more types of violations in that file. So, while egrep.c has the most violations, timer.c has more types of violations. Notice that the text above the treemap indicates the settings used.

Result Treemap:	number of violations : distinct violation type	s \land 🎘 🎕	Options
	Directory Structure		
	fastgrep egrep.c	timer.c	regex strpbrk.c regsub.c
		try.c	
Reconfigure	Result Log Results by File	Results by Check	Result Locator

If you double-click on a file block, you see the Results by File listing with the list of violations for that file expanded.

By default, the treemap is organized using the file structure of the project as architecture nodes. Within the treemap, you can double-click on an architecture node (shown as a gray border around a set of colored blocks) to display only the contents of that node. You can also zoom in by right-clicking on a node and choosing **Drill down** from the context menu.

After drilling down in the architecture, you can use the  $\bigwedge$  icons to **Pop up one level** or **Pop up all levels** the treemap. You can also right-click to use the **Pop up one level** and **Pop up all levels** commands in the context menu.

Codecheck Results Treemap Options	? ×
Group by: Directory Structure  Size Options	
Map Size to: number of violations	
Limit the size of large nodes to 100 🚔 % of the available space	
Color Options	
Map Color to: distinct violation types	
Min Color:	Max Color:
Use Logrithmic Scale	
Cushion Re	store Colors
Generate Treemap	Cancel

Click the **Options** button to modify which metrics are assigned to size and color:

For information about using these fields, see Using the Metrics Treemap on page 240.

Printing and Exporting Results	In the Result Log tab, you can copy the log to your clipboard for pasting by clicking the <b>Copy</b> icon. To save the log to a text file, click the <b>Export</b> icon.
	In the Results by File, Results by Check, and Result Locator tabs, you can click the Print icon to print the currently displayed results. You can click the is <b>Export</b> icon to export the detailed results to an HTML directory, your clipboard, or a text file.
	In the Result Treemap tab, you can click the 🍓 Print icon to print the currently displayed treemap diagram.
	See page 278 for details about the Print Options dialog.
lgnoring Checks and Violations	A number of options let you ignore some CodeCheck violations in all or part of your project. For example, you might want to ignore violations in third-party code used by your project.
	If many violations are detected, you might want to look at the individual checks in the <b>Checks</b> tab to see if you can set options to control the sensitivity of the checks. For example, for the "Magic Numbers" check, you can specify that bitfields can be set to fixed values and you call allow exceptions for values like 0 and 1.
	Wherever you see a violation listed in the results, you can right-click and choose to ignore this violation instance, ignore violations of this check for the specified entity, or ignore violations of this check for the current file or a selected directory level within the project. You can also choose to ignore all violations in a specific file or directory. If you

select a directory, violations in all of its subdirectories will be ignored. You can also click the  $\Theta$  Ignore icon above the results to access the Ignores menu.

2	4 📀	Unused Functions							
1	⊳	C:\Program Files\SciT	C:\Program Files\SciTools\sample\fastgrep\regerror.c						
1	4	C:\Program Files\SciT	C:\Program Files\SciTools\sample\fastgrep\timer.c						
		🚯 😡 Violation: regerro	M	Ignore This Violation	Ctrl+Alt+I				
1	⊳ ⊘	Unused Local Variables	2	langre Violation for Entity: regarder	Ctrl+Alt+O	L			
102	⊳ ⊘	Variables should be com		ignore violation for Entity, regipar	CITHAILTO .	L			
			м	Ignore Violation for File/Directory:	•	L			
			X	Ignore All Violations for File/Directory:	•	L			
				Entity: regnpar	•				
			Ø	Go to Check	Ctrl+Alt+C	l			
			Θ	Go to Ignores List	Ctrl+Alt+J	ŀ			
			۲	Remove Check					

When you choose to ignore a violation, you are first asked to confirm that it is OK to ignore multiple violations of the same type in the same file. Next, you are asked if you want to add a note about this ignored violation. If you click **Yes**, you can type text regarding this violation and why it is ignored.

If you are not sure whether you want to ignore the violation, choose **Go to Check** for more information about the check that was violated.

Violations that you choose to ignore are not listed in the Results by File or Result Locator tabs unless you check the **Show Ignored Violations** box. They are highlighted with a pink background if they are shown. The violation totals in the Results by File tab *do not* include the ignored violations. Totals in the Results by Check tab *do* include ignored violations.

If you choose to ignore any violations, you can use the **Ignores List** tab to find and sort ignored violations. Only one item is listed if you have ignored multiple violations of the same type in a file or directory. You can search this list as you would the Result Locator list. See *Filtering the List* on page 157 for details.

The toolbar above the Ignore List lets you perform the following actions:

Add or edit the text note for the selected violation to be ignored. You can also open the Ignored Violation Note dialog from the right-click menu or by pressing the + key.

Remove the text note for the selected ignored violation. You can also remove the note from the right-click menu or by pressing the - key.

Stop ignoring this violation. You can also open the Ignored Violation Note dialog from the right-click menu or by pressing the Delete key.

Export a list of the violations to ignore to a text file. You are prompted to specify the location and filename. The exported file is a comma-separated values file that lists the full file path, the internal ID string used to identify the entity, the type of violation that occurred, and the check being performed that identified this violation.

Import a list of violations to ignore from a text file. The file must be in the same format exported from the CodeCheck tool.

#### Using CodeCheck Configurations

If you have a set of checks you want to use, you can save that list of checks as a "configuration". Such configurations are stored outside of the project, so that you can use the same CodeCheck configuration with multiple projects.

Note that the set of files to be inspected is not saved as part of a CodeCheck configuration. The most recent set of files used is the default.

To save a configuration, follow these steps:

- 1 In the Checks tab, select the boxes for all the checks you want performed when this configuration is used.
- 2 Click the **Save** icon next to the Configuration drop-down list.

Configuration:	New 🔻		×
----------------	-------	--	---

3 Type a name for your configuration in the dialog, and click OK.

😰 Code Check Save	? ×
New Configuration Name	
Please specify the name of	of the configuration to save.
My checks	
	OK Cancel

You can use a configuration you have saved by selecting it in the **Configuration** dropdown list at the top of the Checks tab.

Another way to run a configuration is from the **CodeCheck > Saved Configurations** menu item, which you can use even if the CodeCheck window is not open.

The **CodeCheck > Re-Run Previous Checks** command runs the most recently selected CodeChecks. If you have made changes to any files, you will be asked if you want to re-analyze the changed files before running CodeCheck.

The CodeCheck > Analyze Changes and Re-Run Previous Checks command runs the Analyze Changed Files command automatically and then runs the Re-Run Previous Checks command.

#### Writing CodeCheck Scripts

CodeCheck scripts are special Perl scripts that let you provide custom checks for verifying your team's coding standards. They can be used to verify naming guidelines, metric requirements, published best practices, or any other rules or conventions that are important for your team.

You can develop these scripts using the Understand Perl API along with a set of special functions designed to interact with the Understand CodeCheck interface.

CodeCheck script files have a \*.upl extension.

To begin writing your own check, follow these steps:

- 1 Choose CodeCheck > Implement Your Corporate Standard from the menus.
- 2 In the web page this command takes you to, save the codecheck\_template.upl to a file with the same name on your computer.
- 3 Edit this template file (with a text editor).
- 4 Modify the name, description, and detailed\_description to match what you plan for this check to do. For example, you could use the following descriptions for a check to make sure lines do not exceed a specified length:

```
# Required - Return the short name of the check
sub name { return "Characters per line";}
```

# Required - Return the short description of the check
sub description { return "Lines should not exceed a set number of characters";}

# Required - Return the long description of the check

sub detailed\_description { return "For readability, lines should be limited to a certain
number of characters. The default is 80 characters per line.";}

5 Modify the test\_language subroutine to test for the desired languages. For example, the following test makes the check apply to C++, Java, and Python. You can look at other scripts in the \conf\plugin\SciTools\Codecheck subdirectory of your installation for more examples.

```
sub test_language {
    my $language = shift;
    return $language =~ /C++|Java|Python/;
    return 1;
}
```

6 If your check should be run on a per-entity basis, return 1 for the test\_entity subroutine. If the check should be run only once per file, return 0 for the test\_entity subroutine. For example:

```
sub test_entity { return 1;}
```

7 If your check should be run only once per project, return 1 for the test\_global subroutine. Otherwise, return 0 for the test\_global subroutine. For example:

```
sub test_global { return 0;}
```

8 If your check requires the user to set options, modify the define\_options subroutine. For example:

```
sub define_options{
    my $check = shift;
    $check->option->integer("charPerLine", "Max Characters per line", 80);
}
Modify the check subroutine to include the check and to signal a CodeCheck
    violation reporting the problem. The following example reports filenames that do not
    begin with a letter character:
```

```
if ($file->name =~ /^ [^a-zA-Z] /) {
    $check->violation(0,$file,-1,-1,"File name does not begin with a letter");
}
```

The following example reports lines longer than the specified maximum number:

```
sub check {
    my $check = shift;
    my $file = shift;
    return unless $file->kind->check("file");
    my $maxChar = $check->option->lookup("charPerLine");
    my $lineNum = 1;
    foreach my $line (split('\n',$file->contents)) {
         my $length = length($line);
         if( $length > $maxChar) {
              $check->violation($file,$file,$lineNum,-1,
                      "$length characters on line(Max: $maxChar)");
         }
         $lineNum++;
    }
}
                         9 Verify that your Perl syntax is correct. The easiest way to do this is to open a
                            command line and run the Perl application that ships with Understand: uperl -c
                            mysample.upl.
                         To learn more, you may want to read about Understand's Perl API. Browsing the
                         CodeCheck scripts that are shipped with Understand can also be very beneficial. If you
                         have questions about CodeCheck scripts, the SciTools website can be a great place to
                         ask them.
Installing Custom
                         You can install a script in Understand by dragging and dropping the script file into the
                         Understand window. You will be asked if you want to install the plugin. Click Install.
Scripts
                         When you install a custom check, you will see a message that identifies the directory
                         where the check was installed. For example,
                         C:\Users\YourName\AppData\Roaming\SciTools\plugin\Codecheck. You can install
                         future checks by copying files directly to this directory.
```

### Chapter 12 Comparing Source Code

This chapter is explains the source-code comparison features provided by Understand.

This chapter contains the following sections:

Section	Page
Comparing Files and Folders	296
Comparing Entities	299
Comparing Text	300
Exploring Differences	301

#### **Comparing Files and Folders**

*Understand* provides a tool for comparing files and folders. To open this tool, choose **Tools > Compare > Compare Files/Folders** from the menus.

Choose	files or folders to compare	?
Left:	C:/project_6_00_00_13	- 🗐 🚺
Right:	C:/project_6_00_00_21	- 🗐 🔒
		Compare Cancel

In this dialog, select a file or folder for the left and right comparison. Both sides should be similar files or similar folders. Click the file button to browse for a file; click the folder button to browse for a directory.

Subdirectories of the directories you choose are also compared.

A quick way to compare two files in a project is to right-click on the name of one file (for example, in the Project Browser) and choose **Compare > Set as Left File/Directory Comparison** from the context menu, right-click another files and choose **Compare > Set as Right File/Directory Comparison**, and then choose **Compare > Compare Files/Directories**.



When begin the comparison, the status bar at the bottom of the *Understand* window shows what is being compared.

The code comparison section of the results window is described on page 301. If you are comparing folders, there is an additional top section that lets you see the folder-level and file-level differences and select individual files whose contents you want to compare.

The comparison uses the following file and folder icons.



By default, all files and folders are listed. You can use the **Show** drop-down to choose whether to restrict the list to showing only:

- **Different:** Show files and folders that either exist in only one version or are different in the two versions.
- Left Only: Show files that are contained in the left version only. All different folders are shown because some may contain files that are only in the left version.



- **Right Only:** Show files that are contained in the right version only. All different folders are shown because some may contain files that are only in the right version.
- **Same:** Show files that are the same in both versions. All folders are shown because some that are different may contain files that are the same.

The **Filter** field lets you type characters you want to match in the directory path or filename. For example, "sim" matches any folders or files with "sim" in their names. All files within folder that match the **Filter** (and the **Show** drop-down setting) are shown. Filtering occurs as you type. If you want to use regular expressions, enable that option by clicking the **Options** button to see that menu. Wildcards are not recognized.

You can change the colors use for folder and file names that differ in the two versions by choosing a color from the **Options** menu and selecting a new color in the color picker.



You can highlight all items that exist in only the left or right version. To do this, first right-click on the file list and choose **Expand All**. Then click the **Select** button and choose either **Orphans left** or **Orphans right**. You will see a warning that some items may have been skipped; this applies only if you did not use **Expand All**.





You can copy folders and files from one side to the other. The copied items overwrite any items with the same names. To copy, first select the items you want to copy. (To copy a folder and its contents, select the folder *and* all the folders and files it contains.) Then click the **Copy/Merge** button and choose either **to the right** or **to the left**. This opens the Copy Files dialog,

which lists the files or folders to be copied. If the list is correct, click **OK**. Click the of icon to undo your last copy/merge change. Click the icon to redo you last undo.

The code comparison section of the results window is described on page 301. By default, the file on the left is in read-only mode, and the file on the right is in read-write mode. You can change the mode for either file by clicking in the file and then clicking "RO" or "RW" in the status bar to toggle the mode.

CodeChecked: 4/24/2012 3:39:25 PM Line: 51 Coli

If you modify one or more files that are listed in the comparison area and then switch to the comparison of another file, you are asked whether you want to save and recompare the files.

You can save changes you make to files in the file and folder comparison. If you have modified a file on the right, you can click the **Save** icon to save that file to its existing location. You can use the **Save As** icon on either the left or right to save a file to a different location. After you save changes, you can click **Rescan** to compare the modified files and folders.

#### **Comparing Entities**

You can compare two entities by choosing **Tools > Compare > Compare Entities** from the menus. You see the Comparison window.

A quick way to compare the code that defines two entities in a project is to right-click on the name of one entity and choose **Compare > Set as Left Entity Comparison** from the context menu, right-click another files and choose **Compare > Set as Right Entity Comparison**, and then choose **Compare > Compare Entities**.

গ	Comp	aris	sonx					1114
Sh	Show: C++ Files							
Filt	er:			1	Filter:			
g; g; in in ∢	zlog.c zlog.h fback.c fhack9 c III		ـــــــــــــــــــــــــــــــــــــ		gzio.c gzjoin.c gzlog.c azlog.h ∢		4	4
(	📄 Opt	ions ge S	) Refresh Save Patch infback9.c:i Sel 📣 Unmerge: 🔶 Prev 🕹 N	in le:	fback9.c xt 6 of	: [ 80	🔶 🔶 infback.c:infback.c 🔲 🕻 ) total diff 🧼 Merge All 🛭 🗇 Unmerg	e,
ſ	17		vindov is a user-su		26		vindov and output	
	18	Ľ	*/		27	t	*/	
I	19		int ZEXPORT inflateBacl		28		int ZEXPORT inflateB	
	20		z stream FAR *strm;		29		z streamp strm;	
	21		unsigned char FAR *wind		30		int windowBits;	
	22		const char *version;		31	_	unsigned char FAR *wind	
	23		int stream_size;		32		const char *version;	
	24	Ę			33		int stream_size;	
	25		struct inflate_stat		34	Ę	- -	
	26		_		35		struct inflate_stat	
	27		if (version == Z_NU		36			Ŧ
	1			_	- 27		if formation 7 and	
	-							
3:	Changed	131	ines on the left to 3 lines on the right	1	ine 14:#	inci fui	lude "Inffrees.h"	
5	Changed	111	ine on the left to 2 lines on the right	ii ji	ine 10.7*	iui /in/	dowBits is in the range 8 15 and win	
6	Changed	121	ines on the left to 3 lines on the right	li	ine 28: in	t Z	EXPORT inflateBackInit (strm, windo	
-	~			nii	10.10			

The middle and lower portions are similar to file comparisons (page 301).

At the top of the comparison is an entity filter (page 129). Select a type of entity in the **Show** drop-down. Then use the lists to select two entities you want to compare. The **Filter** fields let you type a string you want to match anywhere in the entity name. Filtering occurs as you type. Wildcards and regular expressions are not recognized.

If you are comparing an entity other than a file (such as a function), merging changes and saving files in the comparison is not permitted. You can still use the Save Patch button to create a patch file in "unified format".

If you are comparing a file, you can merge changes and use the **Save** and **Sa** 

#### **Comparing Text**

You can compare text that you paste into a window by choosing **Tools > Compare > Compare Arbitrary Text** from the menus. You see a window like this:

م <u>آ</u> ة Tex	1 Text Comparison								
Paste the text to compare into the editors below:									
1	This is a test. 1 This is a text.								
2	Line 2 2 Line 2								
3	Line 3 3								
	4 Line 4								
	OK Cancel								

Paste the before and after text you want to compare into the left and right sides. Then, click **OK** to see the comparison.

হাত	Com	parison <sub>y</sub>	510	Comparisor	1 📈 🐼 d	eflate.c,	20	def	late	.h 🗙 🔎	fast	grep - fast <u>o</u>	grep - C	opy 🗴 🚺	• )	Ě
IJ	Opt	ions	efresh	Save Pate	ch 🛛		7111	-					►LIT			Ð
	Mer	ge Selec	$\langle \rangle$	Unmerge Se	合 Pr	ev	> Nex	ct 2	2 of	2 total (	differe	i 🏠 Merg	ge All	 Un	merge	All
1		This	is a	a test.			1			This	is	a text				
2		Line	2				2			Line	2					
3		Line	3					;								
							- 4			Line	4					
																Ŧ
															Þ	
								-								
1: C	hang	ed 1 line	on the	left to 1 line	on the ri	ght		line	1: T	'his is a	text.					
2: C	hang	ed 1 line	on the	left to 2 line	s on the	right		line	4: L	ine 4						

The text comparison is similar to the comparison between two entities. You can merge and unmerge differences, but cannot save files.

Click the \_\_\_\_\_\_\_\_\_\_ fold icon to hide or view the patch file syntax and/or the list of differences.

A quick way to compare text is to highlight some text, right-click, and choose **Compare** > **Set as Left Text Comparison** from the context menu. Then highlight other text, right-click, and choose **Compare** > **Set as Right Text Comparison**. Finally, choose **Compare** > **Compare Text Selections**.

ŧ

#### **Exploring Differences**

**Code Comparison** 

When you compare items, you see a comparison window. Depending on what you are comparing, you see several of the following areas to help you navigate the differences:

- Changed Entities: This area lets you select files or entities to compare. It differs
  depending on what you are comparing.
- **Code Comparison:** This area allows you to examine the differences in the code. See page 301.
- **Patch File:** The patch area shows the patch file syntax to convert from the left version to the right version. See page 304.
- **Difference List:** This list allows you to select individual differences between two versions. See page 304.

The small fold icon between the areas allows you to close and reopen areas to make more space for the other areas. If you point your mouse to the right or left of either fold icon, you see the pane resize mouse cursor, which allows you to resize the areas as needed.

#### 

The Code Comparison area shows individual differences between versions of an entity. The display is similar to that of common differencing tools.

<b>D</b> pt	tions Refresh Save Patch C:\Program File	s\SciTools	ž 🗇 🗭 C:\Users\Yvonne\Docum ž 🔲	]
🙌 Mer	rge Selec 🜵 Unmerge St 🔶 Prev 🛛 🕂	lext 1 of	i 2 total differe 🍦 Merge All 🛛 🍈 Unmerge	AII
7	#define NSUBEXP 10	7	#define NSUBEXP 16 🍥	*
8 [	<pre>typedef struct regexp {</pre>	8 -	typedef struct regexp {	
9	<pre>char *startp[NSUBEXP];</pre>	9	<pre>char *startp[NSUBEXP];</pre>	
10	<pre>char *endp[NSUBEXP];</pre>	10	<pre>char *endp[NSUBEXP];</pre>	
11	char regstart; /*	11	char regstart; /*	
12	char reganch; /*	12	char reganch; /*	
13	char *regmust; /*	13	char *regmust; /*	
14	int regmlen; /*	14	int regmlen; /*	
15	<pre>char program[1]; /*</pre>	15	<pre>char program[1]; /*</pre>	
16	<pre>L } regexp;</pre>	18	int status; 🔛	
17		17	<pre>} regexp;</pre>	_
18	extern regern *regcomn() ·	18		-
4			b.	

The left side shows the code from the first item you are comparing; the right side shows the code from the second item. The entity path is shown just above the code.

Scrolling of the two versions is synchronized horizontally and vertically. The scrollbar shows the location and size of changed sections of code using the comparison colors.

For certain languages that *Understand* understands—such as C code—you can click the + and - signs in the code to expand and compress code constructs such as if and else statements, functions, extended comments, and so on.

The currently selected difference is highlighted in blue (or bluish purple on some screens) by default. Other differences are highlighted in pink by default.

1431	
1432	return busy == NOTBUSY;
1433	}
1434	L
1435	<pre>void QTrackBackApp::showTotal()</pre>
1436	⊟{
1437	if(!mPopup->isVisible() && mB
1438	⊟#ifdef WIN32

You can use the small fold icon (like the one shown here) between the two code versions to hide the left version of the code temporarily. Or, click the in right arrow next to a code change to do the same thing.

You can edit the source code if you like in the right version of the files. You cannot save code directly to a file. Instead, you can use the Save Patch button to save a patch file or you can copy and paste code with merged differences and edits into another application.

You can select text and copy it to the clipboard. To select text, use the mouse or your keyboard. To select all, press Ctrl+A or right-click and choose **Select All**. To copy text to the clipboard, press Ctrl+C or right-click and choose **Copy**.

As always, right-click on any entity name or other text in the code to see lots of itemspecific options in the context menu.

The status bar at the bottom of the window shows your line location in the source code where you last clicked.

The toolbar at the top of the Code Comparison area contains the following controls:

**Options Options button:** Use the Options drop-down to set the following options:

- **Options->Case Insensitive:** By default, changing the case of a letter is not treated as a difference. For example, if you change "a" to "A", the Difference List shows "No Differences" if that was the only change.
- **Options->Skip Whitespace:** By default, changing the number of spaces or tabs is not treated as a difference. The Difference List shows "No Differences" if only whitespace was changed. You can change this behavior by toggling this option off.
- **Options->Skip Blank Lines:** By default, a different number of blank lines is treated as a difference. You can change this behavior by toggling this option on.
- Options->Files are Unicode: By default, differences are reported only for ASCII files. If Understand says "File is Binary", use this command to turn on Unicode file handling.



- **Options->Hide Common Lines:** By default, all lines in both files are shown. If you check this option, most lines that are the same in both versions are hidden in the left (older) version.
- **Options->Patch lines of context:** The patch area shows the patch file syntax to convert from the left version to the right version. By default, 3 matching lines are shown around a change to provide context. You can choose this option and change the number of lines in the Patch Lines of Context dialog.
- **Color choices:** These options let you change the highlighting in the code comparisons. The Different Word color is an overlay that is combined with the other highlight colors as appropriate.
- **Options->Double Click Merging:** A shortcut for merging is to double-click on a difference in the code. This works only if you enable it here.

The Case Insensitive, Skip Whitespace, and Files are Unicode options are not available if you have made a change to a file.

**Refresh Refresh button:** You can use the Refresh button to update the Difference List at the bottom of the Change Results. This list may become out-of-date if you merge differences or edit the file directly.

Save Patch **Save Patch button:** You can use the Save Patch button to create a patch file in "unified format" (or unidiff). This patch file can be used with the Unix patch tool and other similar programs.



Click the **Prev** and **Next** buttons above the Code Comparison area to jump to another difference between the entities.

	Merge Selected Inmerge Selected You can merge differences into the version of an entity shown on the right. You cannot save		
code directly to a file. Instead, you can save a patch file or copy and paste cod merged differences and edits into another application. To merge differences, for these steps:			
	1 Select a difference in the code or by selecting a line in the Difference List area.		
	2 Click the Merge Selected button. This copies the older (left) version of this difference to the current (right) version of the code. (If you change your mind, click Unmerge Selected.)		
	3 Click the <b>Prev</b> or <b>Next</b> button to move to another difference and repeat the previous step.		
	In the Difference List, merged differences are shown in blue italics. In the code, differences you have merged are highlighted in green. (The currently selected difference is still highlighted in blue/purple, even if it has been merged.)		
	A shortcut for merging is to double-click on a difference in the code if you have enabled <b>Double Click Merging</b> in the Options drop-down.		
	Merge All Unmerge All If you know you want to merge all of the differences, click Merge All. If you want to undo all merges you have		
	made, click <b>Unmerge All</b> .		
Patch File       This area shows the differences in patch file format. Such patch files can be the Unix patch tool and other similar programs. You can hide this area by cl small fold icon above the area.			
	The <b>Patch lines of context</b> command in the <b>Options</b> button menu lets you adjust the number of unchanged lines shown around a difference.		
Difference List	The Difference List area shows a list of the differences in the code shown in the Code Comparison area.		
	1: Changed 1 line on the left to 2 lines on the right line 114: s->stream.transparent = (voidpf)0;		
	2: Deleted 1 line on the right line 117: s->file = NULL;		
	3: Changed 1 line on the left to 1 line on the right line 123: s->crc = crc64(0L, Z_NULL, 0);		

In the Difference List, merged differences are shown in blue italics.

You can hide the Difference List portion of the results by clicking the small **Type and** fold icon below the area. This makes more space for the Code Comparison area.

## Chapter 13 Running Tools and External Commands

This chapter will show you how to configure and use source code editors and other external tools from within *Understand*.

This chapter contains the following sections:

Section	Page
Configuring Tools	306
Adding Tools to the Context Menus	313
Adding Tools to the Tools Menu	314
Adding Tools to the Toolbar	315
Importing and Exporting Tool Commands	316
Running External Commands	317
Using the Eclipse Plugin	318

#### **Configuring Tools**

Select **Tools > User Tools > Configure** from the menus to open the Tool Configurations dialog, where you can configure external tools such as source code editors for use within *Understand*. External tools configured for use will be available for context-sensitive launching. The Tool Configurations dialog provides a number of categories that determine how they are launched.

Edit Entity With	TextPad	New
Edit Location v	vith TextPad	
New User Tool		Delete
		Import
Menu Text	Edit Location with TextPad	
inend reat		
Command	txtpad32.exe	
Parameters	\$CurEntity	
Initial Directory		(
lcon file		
Input	None	•
Output	Capture	T
Understand	perl script	Disable for this project
-Analysis Opti	ons:	
Save All	Rescan 🔲 Analyze Changed File	s 🔲 Analyze All Files
Run Analysis	Options: 🔲 Before User Tool 🔲 Afte	er User Tool
Add to		
	enu 🔽 Main Menu 🖾 Toolbar	

First, use the **User Tools** category of the Tool Configurations dialog to define a command and parameters as follows:

- 1 Click New.
- 2 In the Menu Text field, type the name you want to appear in Understand menus for this tool. You can use variables in the Menu Text. For example, you can use \$CurEntity to put the name of the currently selected entity in the tool name. See Variables on page 308 for a full list of variables.

- 3 If the tool you use is on your executable search path, simply type its name in the **Command** field. If not, use the **Browse** button to specify the full path to its executable.
- 4 In the Parameters field, specify parameters that need to be passed on the tool's command line. See Variables on page 308 for a full list of variables. Variables beginning with \$Cur are current position variables that apply only from a Source Editor window. Variables beginning with \$Decl are declaration variables that apply only when an entity with a declaration is selected. Variables beginning with \$Prompt display a dialog to ask the user for some information. You can use the < sign to separate parameters that need to come from stdin. For example, if the password for a tool needs to come from stdin, use: < \$PromptForPassword</p>

Quote marks in the parameter list are handled the same as quotes in the Microsoft Windows command prompt window and a Linux terminal. This is a change from previous behavior. For example:

Parameter Text	Old Result	New Result
\"some text\"	2 args = { \ , some text\ }	2 args = { "some, text" }
-arg="a b"	2 args = { -arg= , a b }	1 arg = { -arg=a b }
-arg=\$Prompt	2 args = { -arg=, result }	1 arg = { -arg=result }

- **5** In the **Initial Directory** field, specify the directory in which the tool should start running. You can use variables such as \$CurProjectDir in this field.
- 6 In the **Icon file** field, type or browse for a small graphic file to act as the icon for this command. You can choose a BMP, GIF, PBM, PGM, PNG, PPM, XBM, or XPM file.
- 7 Choose the **Input** you want to use for the command. The options are **None** (default), **Selected Text**, and **Entire Document**. The Selected Text and Entire Document options are intended to be used when running a tool from the Source Editor.
- 8 Choose what you want done with the **Output** from the command. Options are:
  - **Discard** the output. This is the default.
  - **Capture** it in a Command Window, which is an area that appears by default near the Information Browser. The command window is reused by default if you run another tool or re-run the same tool. You can force results to go to a new window by unchecking the Reuse box on the command results window(s).
  - Replace Selected Text in the current Source Editor window.
  - Replace Entire Document in the current Source Editor window.
  - Create a New Document in a Source Editor window.
  - Copy to Clipboard so you can paste the results elsewhere.
- 9 Check the **Understand perl script** box if this is a Perl script that uses the Understand Perl API.
- **10** Check the **Disable for this project** box if you do not want this user tool to be available when the current project is open.
- 11 In the "Analysis Options" area, choose actions you would like to be performed before and/or after this user tool has completed its action and returned. These

actions can include saving all files, re-scanning for new files in project directories, analyzing modified files, and analyzing all files.

12 In the "Add to..." area, choose ways you want to access this command in Understand. The Pop Up Menu checkbox adds the tool to the right-click context menu. The Main Menu checkbox adds the tool to the Tools > User Tools submenu. The Toolbar checkbox adds the tool's icon to the toolbar.

To edit settings for an existing tool, select it in the list and make changes as needed. Click **OK** to save your changes. If you want to remove a tool, select it and click **Delete**.

For information about using the **Import** button, see *Importing and Exporting Tool Commands* on page 316.

.....

Variables

Variables beginning with \$Cur are current position variables that apply only from a Source Editor window. Variables beginning with \$Decl are declaration variables that apply only when an entity with a declaration is selected. Variables beginning with \$Prompt display a dialog to ask the user for some information.

Variable	Description	
\$CppIncludes	Lists all of the include directories specified in the Project Configuration. This may be useful, for example, if the tool you want to run is a compiler or linker.	
\$CppMacros	Lists all of the macro definitions specified in the Project Configuration.	
\$CurArchitecture	Name of current architecture.	
\$CurCol	Column position of cursor position in current file.	
\$CurEntity	Full name of selected entity.	
\$CurEntityShortName	Short name of selected entity.	
\$CurEntityType	Type of selected entity.	
\$CurFile	Current file's full path.	
\$CurFileDir	Current file's directory.	
\$CurFileExt	Current file's extension.	
\$CurFileFlatStr	Current file's full path with all directory separation characters (such as / and \) replaced with an underscore (_).	
\$CurFileName	Current file's name not including extension or full path.	
\$CurFileShortName	Current file's name without full path.	
\$CurLine	Line number of cursor position in current file.	
\$CurProject	Current fullname location of opened project.	
\$CurProjectDir	Directory in which the opened project is located.	
\$CurProjectName	Current short filename of opened project (not including extension).	
\$CurReportHtml	Current fullname location of opened project's HTML report.	
\$CurReportText	Current fullname location of opened project's CSV report.	

You can use the following variables in the Command or the Parameter field.

Variable	Description		
\$CurScopeEntity	Scope of current entity.		
\$CurSelection	Currently selected text in the current window (file windows only).		
\$CurWord	The word/text at the current cursor position in the current file window.		
\$DeclCol	Column in which the selected entity was declared, defaults to 1.		
\$DeclFile	Full path name of the file in which the selected entity was declared.		
\$DeclFileShortName	Filename only of the file in which the selected entity was declared.		
\$DeclLine	Line in which the selected entity was declared, defaults to 1.		
<pre>\$DeclScopeEntity</pre>	Name of the entity within which the selected entity is declared.		
\$NamedRoot	Specify \$NamedRoot "namedrootname", where the namedrootname is the actual name of the named root. Note that the named root must be active. This variable can be used in either the Parameters field or the Initial Directory field.		
\$PromptForCheckBox	Prompts user for a true/false value required by the command. A 0 (unchecked) or 1 (checked) is passed to the command in place of this variable. This variable should be followed by a string to be displayed as text next to the checkbox. For example, <pre>\$PromptForCheckBox "Show Debug Text"</pre> displays the following prompt		
	Show Debug Text		
\$PromptForCheckBoxGH	Prompts user with a series of checkboxes displayed in a horizontal group. For example, <pre>\$PromptForCheckBoxGH "Show=Debug Text;Tool Tips;Line Numbers" displays the following prompt. The label ("Show" in this example) is optional. A semicolon must be used to separate items. The text strings for all checked items (separated by spaces) are passed to the command.</pre>		
	Show Debug Text Tool Tips Line Numbers		
\$PromptForCheckBoxGV	Prompts user with a series of checkboxes displayed in a vertical group. For example, <pre>\$PromptForCheckBoxGV "Show=Debug Text;Tool Tips;Line Numbers" displays the following prompt. The text strings for all checked items (separated by spaces) are passed to the command.</pre>		
	Show		
	Debug Text		
	Tool Tips		

Variable	Description		
\$PromptForDir	Prompts user to select a directory and passes the full path as a string. For example, \$PromptForDir "Directory Path=\$CurProjectDir" displays the following prompt with the current project directory as the default. The "" button opens the standard directory selection dialog for your operating system:		
\$PromptForFile	Prompts user to select a file and passes the full path as a string. For example, <pre>\$PromptForFile "Filename=\$CurFile" displays</pre> the following prompt with the current source file as the default. The "" button opens the standard file selection dialog for your operating system:		
	Filename C:\code\pixie_2_2_1\Pixie\src\gui\opengl.h		
\$PromptForPassword	Prompts user for a password. Characters typed in this field are obscured.		
\$PromptForRadioBoxGH	Prompts user for a selection from a set of options displayed horizontally. For example, <pre>\$PromptForRadioBoxGH "Format=PNG; BMP; GIF; JPEG"</pre> displays the following prompt. The text string for the selected item is passed to the command. Format		
	PNG O BMP O GIF O JPEG		
\$PromptForRadioBoxGV	Prompts user for a selection from a set of options displayed vertically. For example, <pre>\$PromptForRadioBoxGV "Format=PNG; BMP; GIF; JPEG"</pre> displays the following prompt. The text string for the selected item is passed to the command.		
	Format		
	• PNG		
	O BMP		
	⊖ GIF		
	◯ JPEG		

Variable	Description		
\$PromptForSelect	Prompts user to select from a drop-down box. For example, \$PromptForSelect "Build Version=Debug;Release;Optimized" displays the following prompt. The text string for the selected item is passed to the command.		
	Build Version Debug Debug Release Optimized		
\$PromptForSelectEdit	Prompts user to select from a drop-down box or edit the text in the box. For example, <pre>SpromptForSelectEdit "Build Version=Debug;Release;Optimized" displays the same prompt as the example for \$PromptForSelect, except that you can edit the string in the box.</pre>		
\$PromptForText	Prompts user for a string required by the command. For example, \$PromptForText "Replace=foo" displays the following prompt and provides a default value. The text provided is passed as a string.		
	Replace foo		

In general, the multiple-selection \$Prompt variables accept strings of the format "label=item1;item2". Any number of items may be separated by semicolons. The item strings for all selected items (separated by spaces) are passed to the command.

The label is optional except in the cases of \$PromptForCheckBox, \$PromptForDir, \$PromptForFile, and \$PromptForText. The default value is optional in the cases of \$PromptForDir, \$PromptForFile, and \$PromptForText.

Prompts are processed after the other types of variables, so you can use other variables in the labels and values. For examples, see \$PromptForDir and \$PromptForFile in the previous table.

In addition, operating system environment variables can be used in prompt syntax. For example, \$PromptForSelect "Dir=\$PATH" presents a drop-down list of all the directory paths in your \$PATH definition.

You can optionally provide the item list in a separate file. In that case, the syntax for most \$Prompt variables is label=@fullpath\_of\_listfile.txt.

You can combine variables to pass all the parameters needed by a command. All prompts are combined into one dialog. For example if the command is "Is", you can use the following parameters to create a dialog that lets you select command-line options for the Is command:

\$PromptForRadioBoxGH "Show option=-A;-a" \$PromptForSelect "Sort=-e;t" \$PromptForCheckBoxGV "Additional flags=-d;-D;-l;-L;-s;-l;-u;-x;-c" \$PromptForDir "Dir:=\$CurProjectDir"

Show option		
⊙ -A ◯ -a		
Sort -e 🗸		
Additional flags		
🗖 -d		
-D		
□ -L		
-s		
🗖 -u		
-×		
-c		
Dir: C:\Program Files\STI\sample\pixie_proj		
OK Cancel		

#### **Adding Tools to the Context Menus**

Once a command is defined in the Tools tab, the **Pop Up Menu** category in the Tool Configurations dialog lists user tools that are currently in the context menu on the left and commands you can add to that menu on the right. (Context menus are sometimes called contextual, shortcut, right-click, or pop-up menus.)

Del Configurations			x
User Tools Pop Up Menu Main Menu Toolbar Export Tools	Pop up menu order Edit Location with TextPad Edit Entity with TextPad	Available Tools          Add         Remove         Move Up         Move Down	
		OK Car	icel

To add a tool to the context menus, select it on the right and click **Add**. To remove a tool from the context menus, select it on the left and click **Remove**.

User tools appear in the context menu in the order they are listed in the left column. Use the **Move Up** and **Move Down** buttons to sort the tools as desired.

The following figure shows a context menu for an entity showing the available external tools.



Tools are active or inactive on the context menu based on the context of the parameters provided to the tool. For example, a source editor that specifies \$DeclFile as a parameter is selectable from the context menu for any entity where the declaration is known, but will not be active for an undeclared entity or when no entity is selected.



#### **Adding Tools to the Tools Menu**

Once a command is defined in the Tools tab, the **Main Menu** category in the Tool Configurations dialog lists user tools that are currently in the **Tools > User Tools** menu on the left and commands you can add to that menu on the right.

P Tool Configurations		<b>X</b>
User Tools Pop Up Menu Main Menu Toolbar Export Tools	Main menu order Edit Location with TextPad Edit Entity with TextPad	Available Tools          Add         Remove         Move Up         Move Down
		OK Cancel

To add a tool to the menus, select it on the right and click **Add**. To remove it from the menus, select it on the left and click **Remove**.

User tools appear on the **Tools** menu in the order they are listed in the left column. Use the **Move Up** and **Move Down** buttons to sort the tools as desired.

	То	ols Window He	p	
		Run Command	V/	। ए३ ए३ 💰 🔍 💷 😑 (
5		User Tools 🔹 🕨		Edit Location with TextPad
2		Editor Macros 🕨		Edit Entity with TextPad 10]
		Scheduler 🕨		Configure
		Compare •	ο,	0, deflate_stored
		Ontions	8,	<pre>4, deflate_fast},</pre>
		options	16,	<pre>8, deflate_fast},</pre>

#### **Adding Tools to the Toolbar**

Once a command is defined in the Tools tab, the **Toolbar** category in the Tool Configurations dialog shows user tools currently in the toolbar in the left box and commands you can add to the toolbar in the right box.

Discrete Tool Configurations			<b>X</b>
User Tools Pop Up Menu Main Menu Toolbar Export Tools	Toolbar order Edit Location with TextPad Edit Entity with TextPad	Av Add Insert Separator Remove Move Up Move Down	OK Cancel

To add a tool to the toolbar, select it on the right and click **Add**. To remove it from the toolbar, select it on the left and click **Remove**.

To add a vertical separator to the toolbar, select the item in the Toolbar order box that should have a vertical line to the right of it. Click **Insert Separator** to add "------" to the list.

Icons for the selected tools appear on the toolbar in the order they are listed in the left column. Use the **Move Up** and **Move Down** buttons to sort the icons as desired.

To change the icon for a particular tool, use the **Icon file** field in the User Tools category.

For example, in the following figure, the first icon is provided by *Understand* to open the Tool Configurations dialog. The second icon is the default icon for a user tool if none is specified.



In this toolbar, two icons have been added for user tools. A separator has been added between them.

¥2	9	•
----	---	---

*Note:* You can control which icons are visible in the main toolbar by right-clicking on the background of the toolbar and checking or unchecking items for the various toolbar sections.

#### **Importing and Exporting Tool Commands**

You can import and export tool commands from files. This makes it easy to share tool commands with co-workers.

1 To export commands, choose Tools > User Tools > Configure from the menus and switch to the Export Tools category. You will see the following dialog.

P Tool Configurations		×
User Tools Pop Up Menu Main Menu Toolbar Export Tools	User Tools to export          Image: Edit Entity with TextPad         Image: Edit Location with TextPad	
	All None	Export to file
		OK Cancel

- 2 Check the boxes next to commands you want to share.
- 3 Click Export to file.
- 4 Choose a location and filename for an initialization file (\*.ini) that contains the selected user tool information.
- 5 Click Save.

To import commands, choose the User Tools category in the Tool Configurations dialog and click the **Import** button. Browse for an initialization file created by another *Understand* user and click **Open**. In the Import User Tools dialog, check the boxes next to the tool commands you want to be available in your copy of *Understand*.

#### **Running External Commands**

The **Tools > Run Command** menu item permits any external command to be run directly from *Understand*. Common commands to invoke are compilers, configuration management tools, and Perl programs written using Understand's API.

The Run a Command dialog looks like this:

🔎 Run a comman	d
Command:	C:\Program Files\SciTools\bin\pc-win64\Perl\dumpvar.pl
Parameters	-
Working Directory	▼ …
Capture Output	STI perl script
	Run Close
Running commands	
	Stop

To run a command, follow these steps:

- 1 Type a **Command** or click ... and browse for a file to run. A number of Perl programs are provided in the *Understand* installation.
- 2 Type any command-line **Parameters** required by the command. Click the right arrow if you want to select one of the special variables. These are listed on page 308.
- 3 Click ... and browse for the directory that should act as the **Working Directory**.
- 4 If you want the output sent to a window in *Understand*, leave the **Capture Output** box checked.
- 5 If you are running a Perl script, check the **STI Perl script** box if this is a script provided by Scientific Toolworks.
- 6 Click Run. The output is shown in a Command Window in *Understand* if you checked the **Capture Output** box. Otherwise, the command runs in the background

and output is shown in the **Running Commands** box. You can select a command from this list and click **Stop** to halt the command.

dest (float *)	
Define: create patches.	cpp{1555}
Modify: create patches.	cpp{1555}
Set: create_patches.cpp	{1555}
i (int) Define: create patches. Modify: create patches. Set: create patches.cpp Use: create patches.cpj	cpp{1515} cpp{1515} {1515} o{1515}

The font used in the Command Window is determined by settings in the Command Window category of the Understand Options dialog, which you can open by choosing **Tools > Options** from the menus. See page 106.

On Unix systems, output to both stdout and stderr are captured.

#### Using the Eclipse Plugin

If you use the Eclipse IDE for code development, you can access a number of *Understand* features within the Eclipse IDE by installing the Understand plugin for Eclipse. These features include the Entity Filter, Information Browser, Metrics, Treemaps, Butterfly graphs and Control Flow graphs. See http://scitools.com/eclipse/ for details on installing this plugin.



#### Chapter 14 Command Line Processing

This chapter shows how to use *Understand* from the command line. Command line processing can be used in a batch file for automatic re-building and report generation of projects.

This chapter describes the "und" command line, which allows you to analyze sources and create *Understand* databases from the command line. In addition, it allows you to generate metrics and reports.

**Note:** The "und" commands were standardized in build 571, and the tool should now be much easier to use. Because of the extensive changes, this new version is not backwards compatible with older versions of und. The old und executable has been renamed "undlegacy". If you have legacy scripts, you should rename the binary run by these scripts in order for them to continue to work.

Most examples in this chapter refer to C/C++ files. However, you can use "und" with any supported language.

This chapter contains the following sections:

Section	Page
Using the und Command Line	320
Using the understand Command Line	328
Using Buildspy to Build Understand Projects	329

#### Using the und Command Line

The command-line tool for creating and building Understand databases is und.

The Understand installer can optionally place the appropriate bin directory in your operating system's PATH definition to simplify running the "und" command line. For example the Windows PATH definition might include the C:\Program Files\SciTools\bin\pc-win64 path.

Und can be run in the following modes:

 Interactive mode: You enter this mode if you simply type und on the command line with no command or text file. While in the interactive shell, settings such as open database are remembered from command to command. This is a good mode to use to test a sequence of commands you want to use in a batch file. You can optionally specify the database to open on the command line to run the interactive shell.

```
c:\Program Files\SciTools\bin\pc-win64>und
Welcome to und. Type "help" for a list of commands. "quit" to quit
und> _
```

• Batch mode: Once you identify a sequence of commands you want to run more than once, you can store them in a text file that you can run in batch mode with the und process command. The text file should contain one command per line. Omit the "und" from each command within a batch file. You can use # to begin comments. For example use either of the following commands to run the sequence in the This.txt file:

und process This.txt und process This.txt MyDatabase.udb

The This.txt file might contain commands similar to these:

• Line mode: You can specify a single command or set of commands on a single command line. You must specify the database to be used on each command line, because it is not remembered from line to line. Commands are run in the order they appear on the command line. The help and list commands cannot be combined with other commands. For example, you could run either of the following commands to create a database, add files, analyze all, and then exit:

```
und create -db c:\myDb.udb -languages c++ add @myFiles.txt analyze -all
und create -languages c++ add @myFiles.txt analyze -all c:\myDb.udb
```

This is the equivalent of running the following set of commands in interactive mode:

create -languages c++ c:\myDb.udb
add @myFiles.txt
analyze -all

Alternately, you could run a sequence of line mode commands like the following:

```
und create -languages c++ c:\myDb.udb
und add @myFiles.txt c:\myDb.udb
und analyze -all c:\myDb.udb
```

In general, und commands are case-insensitive.

Und returns a value of 1 if an error occurred.

Und supports the following options that can be added to any command:

Option	Discussion
-db	Specify the database to use
-quiet	Print only errors. Do not print warnings or informational messages
-verbose	Print extra informational details.

Und accepts a number of separate commands. A different set of options is supported for each of these commands, and separate help is available for each. For example, for help on the add command, type:

und help add

The commands supported by und are as follows:

Option	Discussion	See
add	Adds files, directories, and roots	page 322
analyze	Analyzes the project files	page 325
codecheck	Runs CodeCheck	page 326
create	Creates an empty database	page 322
export	Exports settings, dependencies, or architectures	page 324
help	Gives help information for a command	page 322
import	Imports project settings and architectures	page 324
list	Lists information about the project	page 323
metrics	Generates project metrics	page 326
process	Runs all the commands in a text file in batch mode	page 320
purge	Purges the database	page 325
remove	Removes files, directories, roots, and architectures	page 323
report	Generates project reports	page 325
settings	Sets project settings and overrides	page 324
uperl	Runs Perl scripts	page 326
version	Shows the current software version	page 322

Refer to the sections that follow for details on the commands supported by und.

Getting Help on Und	Since we do frequent builds of <i>Understand</i> , it is likely that this manual may not describe all the options of the "und" command line. We recommend that you check the command-line help. For example, to get details on the report command, type:
	und help report
	You can see the version of <i>Understand</i> for the und command tool by using the following command:
	und version
Creating a New Project	Use the und create command to create a new database (project). Specify the name of the database either with the -db option or as the last parameter. Any settings allowed with the settings command (see page 324) can also be used with create. For example:
	und create -db newDB.udb -languages c++ c#
	und create -open_files_as_read_only on newDB.udb
	For more information, use the following command:
	und help create
Adding Files to a Project	If you have a small number of source files then it may be easiest to just supply their names to the analyzer using the wildcard abilities of your operating system shell. For example: und -db myproject.udb add \usr\myproject und -db myproject.udb add file1.cpp file2.cpp und -db myproject.udb add *.cpp In some cases, there may be too many file locations to use the <i>-add</i> technique. A common command line limitation is 255 characters. A directory with hundreds or thousands of files may easily exceed this limit. If wildcards (for example, proj*.c) do not match the correct list of files or you want more fine-grained/repeatable control over what files are processed, you should create a "listfile". This file must have a format of one filename per line:
	<pre>c:\myfiles\myproject\myproject.c c:\myfiles\myproject\support.c c:\myfiles\myproject\support.c c:\myfiles\myproject\io.c c:\myfiles\myproject\io.h h:\shared\allprojects\file2.c h:\options\file3.c h:\options\file5.c You can then add all of these files as follows: und -db myproject.udb add @myfiles.lis Note that there is no limit on the number of files listed in the list file. Another way to add files to a project is to add the files and file override settings already</pre>
	configured in one database to another database. The command format for this is:

und add source\_project.udb destination\_project.udb

	You can also use the add command to add named roots and Visual Studio projects. Options are available to set the watch behavior, subdirectory adding, the exclude list, file filtering, and languages.
	<b>Exclude strings</b> are processed relative to the top-level directory passed to the add command, and are applied to all files in all subdirectories. The exclude strings are internally processed as follows:
	1 Separate the exclude string into the list of wild cards based on spaces, commas, and semicolons.
	2 For each separate exclude string, replace forward and back slashes with the pattern [/\], which matches either slash.
	<b>3</b> Prepend the absolute path of the top-level directory to the wild card, ensuring that [/\] separates the path from the initial wild card.
	4 Compare the wild card to both the file short name and file long name for every file below that top-level directory. The comparison is done with QRegExp wild card matching.
	Named roots definitions on the "und" command line have the highest precedence. The next precedence is named roots defined as environment variables at the operating system level, and finally by named roots defined in the <i>Understand</i> project configuration. See page 107.
	For more information, use the following command:
	und help add
Removing Items from a Project	Use the und remove command to remove files, directories, Visual Studio files, named roots, and architectures from a project.
	Unless there is a name conflict, the type of item to be removed is automatically detected by und. If there is a conflict, the command defaults to deleting the directory with the specified name. You can use the -file, -vs, -root, and -arch options to override this default.
	For example:
	und remove someFile.cpp myProject.udb
	und remove C:\SomeDirectory myProject.udb
	und -db myProject.udb remove vs1.vcproj vs2.vcproj
	Ear more information use the following command:
	und neip remove
Getting Information about a Project	Use the und list command to list file, setting, architecture, or named root settings in a project. For example:
	und list -tree files myProject.udb
	und list settings myProject.udb
	und list arches myProject.udb
	and fibe foots myrfojeet. aub

There are a number of options for listing settings for the project. You can list all settings, language-specific settings, report settings, metric settings, include directories, macro definitions, and more. For example:

und list -override f1.cpp f2.java settings myDB.udb und list -override @listfile.txt myDB.udb und list -metrics -reports settings myDB.udb und list -all settings myDB.udb und list -lang C++ -macros -includes settings myDB.udb und list -lang fortran settings myDB.udb For more information, use the following command: und help list ..... **Modifying Project** Use the und settings command to modify the settings in a project. You can find the names for each setting by using the following command: Settings und list -all settings myProject.udb For example, the following command adds the specified directory to the list of C/C++ include directories in the project: und settings -c++includesadd c:\myincludes myProject.udb In general, setting names are the same as the field name in Understand, but with spaces omitted. For example: und settings -ReportDisplayCreationDate on myProject.udb und settings -ReportFileNameDisplayMode full myProject.udb und settings -ReportReports "Data Dictionary" "File Contents" myProject.udb und settings -C++MacrosAdd MYLONG="Long Text" myProject.udb und settings -ReportNumberOfPages 250 myProject.udb For more information, use the following command: und help settings Importing into a Use the und import command to import project settings or architectures from an XML Project file. In general, you might use this command when creating a new database to import setting that you have exported from another database. For example: und import settings.xml myNewProject.udb und import -arch myArch.xml myProject.udb For more information, use the following command: und help import **Exporting from a** Use the und export command to export project settings, architectures, or a list of Project dependencies to an XML file. For example, this command exports project settings to an XML file that you can use with the und import command: und export toHere.xml myProject.udb
	This command exports architectures to an XML file that you can use with the und import command:
	und export -arch "Calendar" toHere.xml myProject.udb
	These commands export file, architecture, and class dependencies to a CSV, matrix, or Cytoscape file. Several options are available to control the output of dependencies.
und export -depende und export -depende und export -depende und export -depende	encies file csv output.csv myProject.udb encies class matrix output.csv myProject.udb encies arch myArch csv output.csv myProject.udb encies -col refs -format short file csv out.csv myDB.udb
	For more information, use the following command:
	und help export
Analyzing a Project	Use the und analyze command to run (or rerun) the project analysis.
	When you analyze a project, you have several options. You may re-analyze all files with the -all option (the default), only files that have changed with the -changed option, or a list of files with the -files option. For example:
	und analyze myProject.udb
	und analyze -files @someFile.txt
	und -db myProject.udb analyze -rescan -changed
	und analyze -files file1.cpp file2.cpp myProject.udb
	und -db myProject.udb -rescanwithoutanalyze
	You can scan project directories for new files with the -rescan option. (This is done automatically when you analyze all.)
	If you are doing your first analysis after creating a new project, it doesn't matter which option you choose as it will analyze all files regardless. However, if you are performing this function on a regular basis, you may prefer to do an incremental analysis where only the modified files and any other files dependent on those files are re-analyzed.
	Use the und purge command to remove all analyzed data from the Understand database, leaving only the project definition. This significantly shrinks the udb file size, which you may want to do before sharing the file or backing it up. Running the analyze command will repopulate the project. For example:
	und purge myProject.udb
	For more information, use the following command:
	und help analyze
-	
Generating Reports	Use the und report command to generate reports for the project. This command uses the current report settings, which can be viewed by using the und list command (see page 323), and changed using the settings command (see page 324). For example:
	und list -reports settings myProject.udb und report myProject.udb

Generating Metrics	Use the und metrics command to generate metrics reports for the project. You can generate project metrics (the default), architecture metrics, and the HTML metrics report. For example:		
	und metrics myProject.udb		
	und metrics -arch myArch myProject.udb		
	und metrics -html arch1 arch2 c:\temp myProject.udb		
	This command uses the current metrics settings, which can be viewed by using the und list command (see page 323), and changed using the settings command (see page 324). For example:		
	und list -metrics settings myProject.udb		
	For more information, use the following command:		
	und help metrics		
Using CodeCheck	Use the und codecheck command to run the CodeCheck tool on the project and print the log to the screen. You need to provide the name of a CodeCheck configuration file and an output directory for the reports. For example:		
	und codecheck config.ini C:\temp myProject.udb		
	You can create a CodeCheck configuration file as described in Using CodeCheck Configurations on page 292.		
	Options are provided to specify which files to run the CodeCheck configuration on, whether to show ignored violations, whether to flatten the directory tree, and whether to generate HTML output in addition to the default CSV output. You can also export the list of checks performed and the ignored violations to a file without running the CodeCheck configuration.		
	For example, the following command runs the specified configuration file on the files listed in filelist.txt and generates both the HTML and CSV versions of the results:		
und codecheck -html	-files filelist.txt config.ini C:\temp myProject.udb		
	For more information, use the following command:		
	und help codecheck		
Running Perl Scripts	Use the und uperl command to run Perl scripts from the command line. For example, the following command would run the myScript.pl file with the arg1 space and arg2 arguments passed to Perl:		
und uperl myScript.	pl -quiet "arg1 space" arg2 myProject.udb		
	For more information, use the following command:		
	und help uperl		
	Note that the und uperl command does not support any graphical uperl commands, such as \$ent->draw.		

Creating a List of Files	Where a command accepts a @lisfile.txt for an option, the file must contain one item per line. Full or relative paths may be used. Relative paths are relative to the current directory. A # sign in the first column of a line in the file indicates a comment. If an item has a definition, for example a macro definition, the macro name and its value must be separated by an = sign. For example, <i>DEBUG=true</i> .
	On Unix here are a couple ways to create such a file:
	Use the 'Is' command, as in:
	ls *.c *.h > my_project.txt
	Use the 'find' command to recurse subdirectories, as in:
	findname ``*.c *.h" -print > my_project.txt
	In a Windows command shell:
	Use the dir command with the /b option:
	dir /b *.c *.h > my_project.txt
	<ul> <li>Use the /s option to recurse subdirectories, as in:</li> </ul>

dir /b /s \*.c \*.h > my\_project.txt

## Using the understand Command Line

The *Understand* GUI is launched by the "understand" executable. Normally, you launch this using the shortcuts provided by the installation. If you like, you can modify this using the following command-line syntax.

understand [file\_1 ... file\_n] [-options]

Any filenames listed on the command like are opened along with The *Understand* GUI. For example:

understand source.c source.h -db myproject.udb

The available command-line options (also called command-line switches) are as follows:

Option	Discussion
-contextmenu <i>filename</i> [-line # -col # -text #]	Shows the context (right-click) menu for the specified filename at the mouse location. Optionally shows the context menu for the entity located at -line -col (The -text option provides a name hint for the entity).
-cwd path	Set the current working directory to "path". This takes precedence over the last working directory for a project loaded with -db or -lastproject.
-db <i>filename</i>	Open the project specified by the filename.
-diff left_path right_path	Compare the two specified files or folders as with the Tools > Compare command within <i>Understand</i> .
-existing	Detects any running instance of <i>Understand</i> and sends the command line to that instance.
-importusertools importfile.ini	Import user tool definitions from an initialization file.
-lastproject	Open the last project opened by the application.
-lastproject_cwd	Use the directory of the last opened project as the current working directory.
-new	Force the creation of a new instance of <i>Understand</i> . If you use the operating system to open a file with an extension that opens <i>Understand</i> , by default that file opens in any existing instance. You can use this command-line option to force a new instance to open.
-noproject	Ignore all project load requests on startup. (This also clears the "Open Last Project" application setting.)
-no_splashscreen	Use this option to skip the splash screen when <i>Understand</i> starts up. This setting is stored until you change it in the Tools > Options dialog.
-quiet_startup	Use this option to disable all dialogs and splash screens shown during startup.
-SlowConnect	Allow for a longer timeout period when communicating with the license server.

Option	Discussion
-visit filename [line# column#]	Open the file "filename" in an editor window. Optionally position the cursor at the specified line number and column number in the specified file.
-wait	When used with the -existing option, causes this instance of <i>Understand</i> to block while waiting for the other instance to finish the given command.

## **Using Buildspy to Build Understand Projects**

Buildspy is a tool that allows gcc/g++ users to create an *Understand* project during a build. Buildspy gets lists of files, includes, and macros from the compiler. This can save time and improve project accuracy.

To use Buildspy, follow these steps:

- 1 Change the compiler command from gcc/g++ to gccwrapper/g++wrapper in your makefile or build system.
- 2 Either add the <SciTools>/bin/<platform>/buildspy directory to your PATH definition or use the full path to the gccwrapper/g++wrapper executables in your makefile or build system. On Linux, this might be the /SciTools/bin/linux32/buildspy directory. On Windows, this might be the C:\Program Files\SciTools\bin\pc-win64\buildspy directory.
- **3** Perform a make clean or equivalent command. (This step is optional; Buildspy can be run incrementally to update only the files it is run on.)
- 4 From the directory where your make file is located, run a command similar to the following:

buildspy -db path/name.udb -cmd <compile\_command>

For example:

buildspy -db ~/Documents/MyProject.udb -cmd make

5 When the build has finished running, open the *Understand* project that was created and choose **Project > Analyze All Files**.

The buildspy command sends information from gccwrapper/g++wrapper to Buildspy, which allows it to build a complete *Understand* project. The wrappers then call the corresponding compiler.

The wrappers work with any compiler that has gcc-like syntax. You can use any of the following methods to specify which compilers gccwrapper and g++wrapper should call:

- Use Buildspy's -cc and/or -cxx command line arguments.
- Define the UND\_PBCCCOMPILER and UND\_PBCXXCOMPILER environment variables. These environment variables are checked whenever gccwrapper and g++wrapper are run.
- Edit the configuration file located in \$HOME/.config/SciTools on Linux systems and \$HOME/Library/Preferences on Mac.

## Chapter 15 Quick Reference

This chapter lists of commands provided by *Understand*. These lists provide cross references to information about these commands in this manual.

Since new versions of *Understand* are provided frequently, these lists are subject to change.

Section	Page
File Menu	331
Edit Menu	332
Search Menu	332
View Menu	333
Project Menu	333
Reports Menu	334
Metrics Menu	334
Graphs Menu	335
CodeCheck Menu	335
Annotations Menu	335
Tools Menu	336
Window Menu	336
Help Menu	337

This chapter contains the following sections:.

## File Menu

The **File** menu in *Understand* contains the following commands:

Command	See
New > Project	page 35
New > File	page 181
Open > Project	page 21
Open > File	page 181
Close project_name	page 21
Save project_name As	page 39
Export to Image File	page 276
Save Configuration	page 292
Save filename	page 170
Save filename As	page 170
Save All	page 170
Page Setup	page 190
Print filename	page 190
Print Entity Graph	page 278
Recent Files	page 99
Recent Files > Clear Menu	page 99
Recent Projects	page 99
Recent Projects > Clear Menu	page 99
Exit	page 51

## Edit Menu

The Edit menu in Understand contains the following commands:

Command	See
Undo	page 169
Redo	page 169
Cut	page 168
Сору	page 168
Copy Image to Clipboard	page 276
Paste	page 168
Select All	page 168
Comment Selection	page 179
Uncomment Selection	page 179
Change Case	page 179
Toggle Overtype	page 180
Bookmarks	page 181

### Search Menu

The Search menu in Understand contains the following commands:

Command	See
Find	page 163
Find Previous	page 163
Find & Replace	page 163
Go to Line	page 167
Go to Matching Brace	page 178
Instant Search	page 148
Find in Files	page 150
Replace in Files	page 153
Show Find in Files Results	page 152
Find Entity	page 155

### View Menu

The View menu in Understand contains the following commands:

Command	See
Toolbars	page 162
Browse Mode	page 168
Zoom	page 166
Fold All	page 178
Soft Wrap	page 180
Hide Inactive Lines	page 178
Bookmarks	page 181
Contextual Information	page 164
Entity Filter	page 129
Entity Locator	page 155
Information Browser	page 131
Favorites	page 143
Analysis Log	page 121
Project Browser	page 137
Scope List	page 167
Window Selector	page 160
Previewer	page 177
Dependency Browser	page 140

# Project Menu

The **Project** menu in *Understand* contains the following commands:

Command	See	
Configure Project	page 39	
Rescan Project Directories	page 46	
Analyze Changed Files	page 121	
Analyze All Files	page 121	
Improve Project Accuracy > Undefined Macros	page 123	
Improve Project Accuracy > Missing Includes	page 125	
Improve Project Accuracy > More Information	page 122	
Project Overview Charts	page 246	
Architectures > New Architecture	page 202	
Architectures > Browse Architectures	page 193	
Architectures > Manage Architectures	page 201	

## **Reports Menu**

The Reports menu in Understand contains the following commands:

Command	See
Configure Reports	page 208
Generate Reports	page 210
View Reports > HTML	page 211
View Reports > Text	page 211
Dependency > Architecture Dependencies	page 242
Dependency > File Dependencies	page 242
Dependency > Class Dependencies	page 242
Project Interactive Reports	page 32

### **Metrics** Menu

The Metrics menu in Understand contains the following commands:

Command	See
Metrics Summary	page 232
Browse Metrics	page 233
Export Metrics	page 235
Project Metric Charts > Code Volume	page 237
Project Metric Charts > File Volume	page 237
Project Metric Charts > Average Complexity	page 237
Project Metric Charts > Sum Complexity	page 237
Configure Metric Charts	page 237
Metrics Treemap	page 240

## **Graphs Menu**

The Graphs menu in Understand contains the following commands:

Command	See
Dependency Graphs > By architecture	page 196
Dependency Graphs > Load Saved Dependency Graph	page 197
Project Graphs	page 32
Graphs for selected entity	page 248

## CodeCheck Menu

The CodeCheck menu in Understand contains the following commands:

Command	See	
Open CodeCheck	page 282	
Re-Run Previous Checks	page 292	
Analyze Changes and Re-Run Previous Checks	page 292	
Standards	page 292	
Saved Configurations	page 292	
Implement Your Corporate Standard	page 293	

### **Annotations Menu**

The Annotations menu in Understand contains the following commands:

Command	See
Annotate	page 184
Filter Annotations	page 188
Manage Orphaned Annotations	page 189
Search Annotations	page 187
Annotation Options	page 59
Refresh Annotations	page 59
Display Inline	page 59
Display Hover	page 59
Display Indicator	page 59

## **Tools** Menu

The **Tools** menu in *Understand* contains the following commands:

Command	See
Run Command	page 317
User Tools > <i>tool_name</i>	page 314
User Tools > Configure	page 306
Editor Macros > Record Macro	page 180
Editor Macros > Replay Macro	page 180
Editor Macros > Save Macro	page 180
Editor Macros > Configure Macros	page 114
Scheduler > Scheduled Activities	page 50
Compare > Compare Files/Folders	page 296
Compare > Compare Entities	page 299
Compare > Compare Arbitrary Text	page 300
Options	page 95

## Window Menu

The Window menu in Understand contains the following commands:

Command	See
Close current_file	page 170
Close All Document Windows	page 160
Release Window	page 160
Split Vertically	page 160
Split Horizontally	page 160
Unsplit	page 160
Tile	page 160
Cascade	page 160
Predefined Windows Layouts	page 160
<open file="" list="" source=""></open>	page 160
Windows	page 160

# Help Menu

The **Help** menu in *Understand* contains the following commands:

Command	See
Help Content	page 16
Key Bindings	page 104
Example Projects	page 16
PERL API Documentation	page 32
Python API Documentation	page 32
Frequently Asked Questions	page 16
Getting Started	page 21
Licensing	page 14
View SciTools Blog	page 16
Check for Updates	page 21
About Understand	page 16

# Index

### Symbols

^ in regular expressions 158 : in F77 identifiers, Fortran 82 ? in regular expressions 158 ? wild card 158 . in regular expressions 158 " prefixing octal constants or string literals, Fortran 82 "" surrounding normal includes 71 [-] in regular expressions 159 [] in regular expressions 159 [^] in regular expressions 159 \* comments field 82 \* in regular expressions 158 \* wild card 158 /\* ... \*/ C-style comments 68, 82 \ in regular expressions 159 \< in regular expressions 158</p> > in regular expressions 158 # comment in command line file 327 + expanding tree in Information Browser 132 + in regular expressions 158 <> surrounding system includes 71 = macro definition in command line 327 | in regular expressions 159 \$ in regular expressions 158 \$ prefixing external tool parameters 307, 308

### A

About Understand option, Help menu 16 absolute portability mode 47 Activate when Control key is pressed field, Browse Mode Editor options 116 actual parameters, relationship to formal parameters, Ada 62 Ada configuration for 61 macro definitions 63 versions supported 15 Ada category, Project Configuration dialog 61 Add found include files to source list field, C++ Includes 70, 78 Add found system include files to source list field, C++

Add found system include files to source list field, C-Includes 70, 78 -addDir option, und command 322 -addFiles option, und command 322 Adobe Acrobat, saving graphical views to 279 Advanced category, Editor options 111 Aggregate Nodes by options 273 Alerts category, User Interface options 100 Allow 82 Allow Colons in Names field, Fortran 82 Allow C-style comments field, Fortran 82 Allow embedded SQL field, Pascal 88 Allow function declaration without parentheses field, Fortran 82 Allow interactivity during intensive processing field, General options 97 Allow Nested Comments field, C++ 68 Allow parameter declaration without parentheses field, Fortran 82 Allow PreExpansion field 108 Allow quote in octal constants field, Fortran 82 Alternating row colors field 102 Analysis Log option, View menu 122 Analysis Log window 122 reopening previous analysis Log 161 saving to text file 122 Analysis Options area 307 Analyze All Files option, Project menu 51, 121 Analyze category, Understand Options dialog 105 Analyze Changed Files option, Project menu 51, 121 -analyze option, und command 325 -analyzeFiles option, und command 325 analyzing projects 121 after changing configuration of 40 beep on completion of 105 on command line 325 scheduling 51 and operators, including in strict complexity, Ada 62 angle brackets (<>) surrounding system includes 71 Animate windows/drawers field, User Interface options 98 Annotations category, Project Configuration dialog 59 Annotations menu 335 anti-virus software, turning off while generating reports 210 Apache Lucene syntax 148 APIs 32 Append the names of externally linkable entities with field

C++ 69 Fortran 82 Application font field, General options 96 Architect Manager 195, 201 Architecture Browser 30, 193 duplicating architectures 195, 202 graphical views in 194, 196 metrics in 194, 200 opening Architect Manager from 195, 201 opening Architecture Builder from 195 renaming architectures 195 right-clicking in 194 XML listing in 195 Architecture Builder 195, 205 Architecture Graph View 258 Architecture Wizard 203 architectures 19, 192 auto-architectures 193 creating 202 duplicating 195, 202 editing. See Architecture Builder exporting dependency list 242 graphical views of 194, 196 hierarchies of, exploring 193 hierarchies of, graphical view of 258 list of 193 listing 30 managing. See Architect Manager metrics about 194, 200 metrics graphs about 237 navigating 30 renaming 195, 202 XML listing of 195, 206 arrows, for Information Browser history 136 ASCII text. See text ASP style tags 94 assembly configuration for 65 embedded in C code 69, 73 include files, directories for 65 Assembly category, Project Configuration dialog 65 asterisk (\*) in regular expressions 158 wild card 158 Auto Category, C++ Includes 71 auto-architectures 30, 193, 194 Auto-complete fields, Advanced Editor options 112

Auto-indent fields, Advanced Editor options 113, 114 Automatic compool file field, JOVIAL 86 Average Complexity metrics graph 237

#### В

background processing, interactivity during 97 backslash (\) in regular expressions 159 base classes count of 227 displaying 254, 256, 261 Base Classes View 254, 256 beep on analysis completion 105 bitmaps, saving graphical views as 276 black text 115 blank lines in Entity Comparison area 303 Blank metric 229 blocks expanding and collapsing 166, 178 in Filter Area 25 blue italic text 115 blue text 115 bookmarks creating 181 list of 162, 182 navigating 181 Bookmarks option, Edit menu 181 Bookmarks option, View menu 162, 182 Boolean searches 149 braces, matching. See brackets brackets in regular expressions 159 matching 178 Browse Architectures option, Project menu 193 Browse Metric Charts option, Metrics menu 241 Browse Metrics option, Metrics menu 231, 233 Browse Mode option, View menu 168 Browse Mode, Source Editor 168 Browser Area 20 Build Log 34 Buildspy 329 Butterfly View 254, 255 Butterfly-Dependency graph 196

#### С

C/C++

API for custom reports 32, 213, 231 configuration for 68, 75, 79 include files, auto including 71 include files, directories for 70, 77 include files, ignoring 71 macro definitions 72, 79 preprocessor directives 72 strict analysis 75, 79 versions supported 15 C# configuration for 80 reference files, importing 80 versions supported 15 C# category, Project Configuration dialog 80 C++ (Strict) category, Project Configuration dialog 75, 79 C++ category, Project Configuration dialog 68 caching include files, C++ 69 Calendar auto-architecture 30, 193 Called by menu, graphical views 264 Called By View 254, 255 Calls View 254, 255 capitalization, of selected text 179 Captured output font field, Command Window options 106 caret (^) in regular expressions 158 Caret Line field, Editor options 110 Cascading Style Sheet 210 case changing for selected text 179 of displayed entity names, Ada 62 of displayed entity names, Fortran 83 of displayed entity names, JOVIAL 87 of displayed entity names, Pascal 89 of displayed entity names, PL/M 90 of externally linkable entities, Ada 62, 87 of externally linkable entities, Fortran 82 of externally linkable entities, Pascal 89 of identifier names, Fortran 82 Case of externally linkable entities field Ada 62, 87 Fortran 82 Pascal 89 Case sensitive identifiers field, Fortran 82 case sensitivity of Entity Comparison area 303 of Fortran identifiers 82

CBO (Count of Coupled Classes) metric 227 Change Case menu option 179 Change Case option, Edit menu 179 Change Log 34 Changed Entities area, Change Results window 301 character strings, red text for 115 Check for Updates option, Help menu 22 Child Lib Units View 254 children of entity 254 CIS. See Contextual Information Sidebar Class and Interface Cross-Reference report 216 Class Declaration View 258, 261 class diagram 260 .class files 84 Class Inheritance View 254 Class Metrics report 227 Class OO Metrics report 227 Class Paths category, Java 84 classes base classes for 227, 254, 256, 261 cohesion of 227 coupled 227 declaration for 261 derived 227, 254, 257 exporting dependency list 243 extended by other classes 254, 257, 265 extending other classes 265 as hexagons, in graphical views 263 implemented 265 implemented by 265 information about 258 inherited from other classes 254 listing in Filter Area 25 metrics about 227 providing without source code, Java 84 reports about 216, 220, 227 clipboard 302 copying graphical view to 276 copying text from Source Editor to 168 Close All Document Windows option, Window menu 160, 170 Close option File menu 22 Window menu 160, 170 Close Selected Window(s) command 162 cluster graphs 272 COBOL

configuration for 66 copybook files, directories for 67 COBOL category, Project Configuration dialog 66 Code metric 229 Code Volume metrics graph 237 CodeCheck 280 checks 283 configuration 285, 292 ignoring violations 290 result log 285 running 282 scripts 293 CodeCheck menu 335 cohesion for class data members 227 colon (:) in F77 identifiers, Fortran 82 Color Mode field, Advanced Editor options 111 colors in Entity Comparison area 303 in file/folder comparisons 297 of highlighted differences in Entity Comparison area 302 in HTML reports 210 of merged differences in Difference List area 304 of merged differences in Entity Comparison area 304 for printing source code 111 of rows in User Interface, alternating 103 in Source Editor, customizing 166, 115 in Source Editor, default 115 column truncation Fortran 82 **JOVIAL 86** command line. See und command; understand command command renames, reports about 221 Command window 317 Command Window category, Understand Options dialog 106 commands, external. See external tools Comment metric 229 Comment Selection menu option 179 comments adding or removing 179 associating with entities, Ada 62 associating with entities, C 76 associating with entities, C++ 69 associating with entities, Java 84 blue italic text for 115

C-style 68 nested, C++ 68 companion file 181 Compaq Pascal 88 Compare Arbitrary Text option, Tools menu 300 Compare Entities option, Tools menu 299 Compare files by content field 78 Compare Files/Folders option, Tools menu 296 compilation environment, Ada 62 compilation unit Child Library Units of 254 declaration nesting of program units in 254 entities called by 254 instantiation tree for 254 With By relationships of 254, 271 With relationships of 254 compiler COBOL 66 compared to Understand 34 Compiler field C++ 68 PL/M 90 Compiler Include Paths field, C++ 68 complexity exception handlers included in, Ada 62 FOR-loops included in, Ada 62 metrics graphs about 237 strict, and/or operators in, Ada 62 Component field, Key Bindings options 104 compool file, JOVIAL 86 Configure category, Understand Options dialog 106 Configure Metric Charts option, Metrics menu 231, 237 Configure option, Project menu 39 Configure Reports option, Reports menu 208 constants displayed in graphical views 264 reports about 224 Constants menu, graphical views 264 contact information 15 Context Browser, Contextual Information Sidebar 164 -contextmenu option, understand command 328 Contextual Information option, View menu 164 Contextual Information Sidebar (CIS) 164 Contiguous Selection field 102 Control Flow View 258, 262 **!COPY** directives, directories to search, JOVIAL 87 Copy category, JOVIAL 87

Copy option, Edit menu 168 Copybook category 67 copybook files adding to project, COBOL 67 copying text 302 rectangular area 168 Count and/or operators in strict complexity field, Ada 62 Count exception handlers in complexity field, Ada 62 Count for-loops in complexity field, Ada 62 Count of All Methods metric 227 Count of Base Classes metric 227 Count of Coupled Classes metric 227 Count of Derived Classes metric 227 Count of Instance Methods metric 227 Count of Instance Variables metric 227 Count of Methods metric 227 coupled classes 227 Create and cross-reference record object components field. Ada 62 Create implicit special member functions field 69 Create references in inactive code field, C++ 69 Create references in inline assembly, C++ 69 Create references to local objects field, C++ 69 Create references to macros during macro expansion field, C++ 69 Create references to parameters field, C++ 69 Create relations between formal and actual parameters field. Ada 62 Crossing layout option 267 cross-reference reports 214 cross-referencing record object components, Ada 62 CSS files 15 C-style comments in Fortran 82 nesting of, allowing 68 CSV file exporting dependencies to 242 exporting metrics to 51, 231, 235 exporting project overview charts to 247 Ctrl+Alt+F keystroke, find and replace text 163 Ctrl+F keystroke, find text 163 Ctrl+j keystroke, jump to matching bracket 178 Ctrl+right-click keystroke creating new windows 23, 127 Ctrl+Shift+j keystroke, select text in brackets 178 \$Cur variables 307.308 Cut option, Edit menu 168

-cwd option understand command 328 Cyclomatic complexity 222, 229 Cytoscape installation location 108 XML output 243 Cytoscape XML 242, 243

#### D

dashed outline shapes, in graphical views 263 Data Dictionary report 214 Data Members View 258 database 19, 34 changes to format of 34 creating on command line 320 file extension for 19, 34 multi-user read/write access for 34 See also project Date Format field 112 -db option understand command 328 **DEC Pascal 88** \$Decl variables 307, 308, 309 Declaration File View 258, 260 Declaration Tree report 219 **Declaration Tree View 254** Declaration Type View 258, 261 Declaration View 258, 259 declaration views calling methods displayed in 264 constants displayed in 264 default members displayed in 264 extended by classes displayed in 265 extended classes displayed in 265 external functions displayed in 265 file dependencies displayed in 264 globals displayed in 265 header files include by's displayed 266 implemented by classes displayed in 265 implemented classes displayed in 265 imported entities displayed in 266 include files displayed in 266 inherited entities displayed in 266 invocations displayed in 266 local items displayed in 268 members displayed in 268 objects displayed in 268

operators displayed in 268 private members displayed in 269 protected members displayed in 269 public members displayed in 269 rename declarations in 269 static functions displayed in 270 types displayed in 270, 271 used-by items shown in 271 variables displayed in 271 With By relationships displayed in 271 With relationships displayed in 271 See also graphical views declarations implicit, report about 223 local, including in database, C++ 69 Declared In View 254 Default Members menu, graphical views 264 Default style field, Editor options 109 default working directory 97 defines. See macros Delphi Pascal 88 Depended On By graph 196, 254 dependencies browsing 140 exporting 141, 242 file 264 Dependency Browser 140 Dependency category, Understand Options dialog 108 Dependency Graph 194 Dependency Graphs option, Graphs menu 196 Dependency menu 242 Dependent menu graphical views 264 Dependent Of menu, graphical views 264 Depends On graph 196, 254 Depth menu, graphical views 264 derived classes 227, 254, 257 Derived Classes View 254, 257 DFM converter exe field 88 -diff option, understand command 328 Difference List area, Change Results window 301, 304 Dim highlight color field 103 directives, C++ 72 directories adding to project 43 comparing 296 copying 297

deleting from project 44 overriding settings for 45 watched, scanning 46, 51 watched, setting 44, 45 Directory Structure auto-architecture 30, 193 Display entity names as field Ada 62 Fortran 83 JOVIAL 87 Pascal 89 PL/M 90 DIT (Max Inheritance Tree) metric 227 Dock Window Layouts field, User Interface options 98 docking windows 18 Document Area 20 documentation 16.22 dollar sign (\$) in regular expressions 158 prefixing external tool parameters 307, 308 DOS line termination style for reports 210 for saving source files 110 DOT files, saving graphical views as 277 double quotes. See quotes double-click merging, in Entity Comparison area 303, 304 double-clicking in Bookmarks area 182 column header dividers 155 entities in Entity Filter 134 entities in graphical view 250 entities in Information Browser 134 in Exploring View 139 in Find Results window 152 messages in Analysis Log window 122 in Project Browser 233 in Source Editor 29 drawers 18.98 Drill down command 242 drop-down menu See right-clicking .dsp file extension 57 .dsw file extension 57 Duplicate Architecture menu option 195 duplicate references, C++ 69 Duplicate Subtrees menu, graphical views 264

#### Ε

Eclipse IDE 318 Edges Shown options 273 edges, in dependency graphs 198 Edit Architecture menu option 195, 205 Edit Graphic Filters menu option 252 Edit menu 332 Bookmarks option 181 Change Case option 179 Copy option 168 Cut option 168 Select All option 168 Toggle Overtype option 180 Editor category, Understand Options dialog 109 Editor Macros option. Tools menu 180 editor windows, saving automatically 96 editor, external 117 editor, source, See Source Editor Emacs editor 117 embedded SQL, in Pascal 88 Enable Editor Tooltips field 116 Enable permissions checking for NTFS filesystems field 97 encoding formats 49, 109 entities 19 comments associated with, Ada 62 comments associated with, C 76 comments associated with, Java 84 comparing two entities 299 comparisons between versions of 301 components of 258 current, information about 164 declaration structure for 258, 259 displaying source for. See Source Editor favorites of, displaying 143 favorites of, marking 142 full name of, displaying 130 graphical views of. See graphical views hierarchy of, exploring 139, 164 hierarchy of, graphical view 249, 254 information about. See Information Browser list of, alphabetic 214 listed in Entity Filter 129 listed in Entity Locator 26, 155 listed in Filter Area 25 metrics for. See metrics

names of, formatting for reports 55 as rectangles, in graphical views 263 references for 135 relationships between 19 renamed 258 reports about. See reports sorting of 129 structural information about, graphical view of 250, 258 unknown 271 unresolved 271 XML listing of 195 See also specific entities Entity Comparison area, Change Results window 301 Entity Filter 129 displaying information about entities in 131 displaying source of selected entity 134 entities not listed in 26, 155 iumping to entities in 25 location of 20 right-clicking in 24 Entity Graph option, Window menu 250 entity index, for HTML reports 211 Entity Locator 26, 155 column headers in, customizing 156 columns in, hiding and reordering 156 columns in, resizing 155 filtering by selection 157 filtering manually 158 filtering with regular expressions 158 filtering with wildcards 158 length of names in 155 opening 155 right-clicking in 26, 155 right-clicking on column header 156, 158 right-clicking on entities 157 sorting entities in 156 Entity Locator option, View menu 26, 155 entity\_index.html file 211 environment variables using in include paths, assembly 66, 67 using in include paths, C++ 70, 77 equal sign (=) in macro definition in command line 327 errors displaying from Analysis Log window 122 parse errors, prompting for, Ada 63 parse errors, prompting for, Fortran 83

Essential metric 229 Example Projects option, Help menu 16 Exception Cross-Reference report 218 exceptions handlers for, including in Ada complexity 62 reports about 218 -existing option, understand command 328 Expand Recursive Nodes menu, graphical views 265 Expand Repeated Nodes menu, graphical views 265 Expand/Collapse code snippet field 102 Exploring view 139 Export Dependency CSV 242 Export Metrics option, Metrics menu 51, 231, 235 exporting tool commands 316 Extend Tree report 220 Extended By menu, graphical views 265 Extended By View 254, 257 Extends menu, graphical views 265 extensions. See file extensions external editor 117 External Editor category, Editor options 117 External Functions menu, graphical views 265 external tools adding to Right-click Menu 308, 313 adding to toolbar 315 adding to User Tools menu 314 commands for, importing and exporting 316 commands for, running 317 configuring 306 editor 117 externally linkable entities case of, Ada 62, 87 case of, Fortran 82 case of, Pascal 89 prefix for, C++ 69 prefix for, Fortran 82 suffix for, C++ 69 suffix for, Fortran 82 Externally Modified Files field, Editor options 110 Extract Function refactoring 174 Extract Temp refactoring 176

#### F

F5 key, Find in Files option 150 FAQ option, Help menu 16 favorites 27

displaying 143 marking entities as 142 Favorites option, View menu 27, 143 File Average Metrics report 229 File Contents report 215 file extensions configuring for project 48 for database 19, 34 for MSVC project files 57 File Information, Contextual Information Sidebar 164 File menu 331 Close option 22 New > File option 181 New > Project option 35 Open > File option 181 Open > Project option 22 Page Setup option 190 Print Entity Graph option 278 Print option 190 Recent Projects option 22 Save All option 170 Save option 170 File Metrics report 228 File Mode field, Editor options 109 File Options category, Project Configuration dialog 49 file permission checking 97 File Sync box 134 File Types category, Project Configuration dialog 48 File Volume metrics graph 237 Filename menu, graphical views 265 filenames in graphical views 265 for reports 211 in title areas 98 files comparing 296 copying 297 exporting dependency list 242 Information Browser for 164 location included in listing of 130 searching 27, 150 searching and replacing in 153 toolbar for 162 See also database; header files; include files; MSVC project; project; source files Files category, Project Configuration dialog 42 Filter Area 25

See also Entity Filter; Project Browser Filter By Selection menu option 157 Filter field, Entity Filter 130 filters for graphical views 252 See also Entity Filter; Entity Locator Find dialog 163 Find Entity option, Search menu 155 Find in Files dialog 20, 27 Find in Files menu option 27, 150 Find option, Search menu 163 Find Results window 27, 152 fixed file format, Fortran 81 flow chart symbols 263 flow charts 258, 262 Fold All option, View menu 178 fold icon 301 folders. See directories folding code 178 font in Command window, changing 106 for printing source code 111 for Source Editor windows 109 in Source Editor windows 183 in Understand, changing 96 Font Size field, Advanced Editor options 111 FOR-loops, including in complexity metrics, Ada 62 formal parameters, relationship to actual parameters, Ada 62 Format field, Fortran 81 Fortran configuration for 81 extensions supported 15 include files 83 macro definitions 83 reports showing non-standard extension usage 223 versions supported 15 Fortran category, Project Configuration dialog 81 Fortran Extension Usage report 223 frames in windows, sliding 18 Frameworks Category, C++ Includes 78 free file format, Fortran 81 function declarations without parentheses, Fortran 82 Function Pointer menu, graphical views 265 functions external 265 in graphical views 269

listing in Filter Area 25 metrics about 229 reports about 215, 229 static 270 *See also* invocations; procedures; program units fuzzy search 149

#### G

gcc build 329 General category, Understand Options dialog 96 Generate Reports option, Reports menu 210 Generic Instantiation report 221 generic unit instantiation hierarchy for 254 instantiation tree for 254 Getting Started dialog displaying at startup 96 opening 22 Getting Started option, Help menu 16, 22 Getting Started tab 21 Globals menu, graphical views 265 Go To Line dialog 168 Go to Matching Brace option, Search menu 178 Graph Architecture 194, 196 Graph Architecture graph 196 Graph Architecture View 258 Graph Customizer pane 196 Graph View of graphs 246 Graph View, for metrics graphs 239 Graphic Filter dialog 252 graphical user interface (GUI), parts of 20 graphical views 31, 246 of architecture 194, 196 browsing 250 calling methods displayed in 264 constants displayed in 264 copying to clipboard 276 default members displayed in 264 diagram of 20 displaying 248 entity name truncation for 270 expanding or contracting nodes in 251 extended by classes displayed in 265 extended classes displayed in 265 external functions displayed in 265 file dependencies displayed in 264 filename display options 265

filtering 252 fullnames displayed in 268 function pointers displayed in 265 functions displayed in 269 globals displayed in 265 Graph View of 246 header files include by's displayed 266 hierarchy levels displayed in 268 hierarchy views 249, 254 implemented by classes displayed in 265 implemented classes displayed in 265 imported entities displayed in 266 include files displayed in 266 inherited entities displayed in 266 intrinsic functions displayed in 266 invocations displayed in 266 layout configuration for 252, 263, 267 local items displayed in 268 members displayed in 268 multiple subtrees displayed in 264 objects displayed in 268 open, list of 250 operators displayed in 268 parameters, displaying in 269 path highlighting in 251 printing 251, 278 private members displayed in 269 procedures displayed in 269 protected members displayed in 269 public members displayed in 269 rename declarations in 269 reusing 253 saving 251, 275 saving as PDF 279 scrolling in 250 sorting entities in 270 spacing entities in 270 SQL entities shown in 270 static functions displayed in 270 structure views 250, 258 synchronizing with other windows 253 Table View of 247 text size of 269 types displayed in 270, 271 unknown entities, displaying 271 unresolved entities, displaying 271 used-by items shown in 271

uses items shown by 271 variables displayed in 271 With By relationships displayed in 271 With relationships displayed in 271 zooming 251 See also structure views; hierarchy views Graphical Views menu option 194, 196, 248 Graphs category, Understand Options dialog 118 Graphs for option, Graphs menu 196, 248 Graphs menu 335 Dependency Graphs option 196 Graphs for option 196, 248 Project Graphs option 32, 213 graphs, of metrics 237 gray background for text 115 gray shapes, in graphical views 263 green text 115 GUI (graphical user interface), parts of 20 GVIM editor 117

#### Η

header files 266 help 16, 22 Help menu 337 About Understand 16 Check for Updates option 22 Example Projects option 16 FAQ option 16 Getting Started option 16, 22 Help Content option 16 Key Bindings option 16, 180 Licensing option 14 Perl API Documentation option 16, 32, 231 Python API Documentation option 16, 32 View SciTools Blog option 16 -help option, und command 322 hexagons, in graphical views 263 Hide Inactive Lines option, View menu 178 hiding common lines, in Entity Comparison area 303 hierarchy views 31, 249 layout of 267 list of 254 multiple subtrees displayed in 264 number of levels in 268 parameters, displaying in 269 See also graphical views highlighting full line at cursor 110

history Information Browser 132, 136 Source Editor 161 toolbar for 162 Horizontal Non-Crossing layout option 267 HTML colors in reports 210 entity index in reports 211 exporting metrics to 51, 231, 234 files 15 generating reports as 32, 54, 208, 210 viewing reports as 211 Hyper Grep. See Find in Files dialog hyphen (-) collapsing tree in Information Browser 132

#### I

IB. See Information Browser IFANIN (Count of Base Classes) metric 227 Ignore category, C++ Includes 71 Ignore directories in include names field 71, 78 Ignore Parent Overrides field 46 Implementation fields field, JOVIAL 86 Implemented By menu, graphical views 265 Implements menu, graphical views 265 implicit special member functions, C++ 69 Implicitly Declared Objects report 223 import files adding to project, Python 92 Import report 221 imported entities, displaying in graphical views 266 importing tool commands 316 Imports category Python 92 Imports menu, graphical views 266 -importusertools option, understand command 328 inactive code, cross-reference information for, C++ 69 inactive lines, hiding 178 Include By View 256 Include File Cross-Reference report 217 include files adding as source files, assembly 65 adding as source files, C++ 70, 77 adding as source files, Fortran 83 adding as source files, PL/M 91 adding before each project file, C++ 71 adding in bulk, assembly 66, 67

adding in bulk, C++ 70, 77 adding in bulk, Fortran 83 adding in bulk, PL/M 91 adding to project, assembly 65 adding to project, C++ 70, 77 adding to project, COBOL 67 adding to project, Fortran 83 adding to project, PL/M 91 compiler path for, C++ 68 displayed in graphical views 266 environment variables in paths for, assembly 66, 67 environment variables in paths for, C++ 70, 77 hierarchy of, for source files 254, 256 ignoring during analysis, C++ 71 overriding MSVC project settings for 58 Pascal search path 90 replacement text for, C++ 72 replacement text for, Fortran 83 replacement text for, PL/M 91 reports about 217 system include files, C++ 71 Include line numbers in rich text field 112 Include View 254, 256 IncludeBy View 254 Included By menu, graphical views 266 Includes category C++ 70, 77 Fortran 83 PL/M 91 Includes menu, graphical views 266 Indent field, Editor options 110 indentation 113, 114 fixing 180 Information Browser (IB) 25, 28, 131 architecture information in 193 copying information in 132, 136 displaying from entity in Entity Filter 131 displaying source of selected entity 134 entity information displayed by, choosing 133 expanding and collapsing the tree 132 for current entity 164 for current file 164 history for 132, 136 location of 20 metrics in 135, 231 multiple occurrences open 134 References in 135

right-clicking in 24 saving information in 132, 136 searching 133 synchronizing 134 Information Browser option, View menu 131 Inherited By menu, graphical views 266 Inherits menu, graphical views 266 initialization files 22 Inline Function refactoring 173 Inline Temp refactoring 175 Insert Spaces Instead of Tabs field, Editor options 110 instance methods 227 instance variables 227 Instant Search 148 Instant Search menu option 148 Instantiated From View 254 instantiation of compilation units 254 of generic types 254 of generic units 254 reports about 221 Instantiations View 254 interactivity during background processing 97 interfaces listing in Filter Area 25 reports about 216 Internal Dependencies graph 196 interrupt handlers, listed as unused program units 225 intrinsic functions, parsing, Fortran 82 Intrinsic menu, graphical views 266 Intrinsics file field, Fortran 82 Invocation Tree report 220 Invocation View 254 invocations reports about 220 views of 254, 255, 258, 259 Invocations menu, graphical views 266

#### J

.jar files 84 Java configuration for 84 versions supported 15 Java category, Project Configuration dialog 84 JavaScript files 15 JDK, versions supported 15 JNI external entities, Java 84 JOVIAL configuration for 86 directories for !COPY directives 87 versions supported 15 JOVIAL category, Project Configuration dialog 86 JPEG format, saving graphical views as 276 jQuery analysis 94 Jump to Matching Brace menu option 178 Jump to Matching Directive menu option 178

#### Κ

Key Bindings category, Understand Options dialog 104, 180 Key Bindings option, Help menu 16, 180 keyboard mappings 104 Keyboard Scheme field, Key Bindings options 104 keywords, blue text for 115 KNI external entities, Java 84

#### L

Language auto-architecture 30, 193 Languages category, Project Configuration dialog 41 -lastproject option, understand command 328 -lastproject\_cwd option, understand command 328 Layout menu, graphical views 267 layout, for graphical views 267 LCOM (Percent Lack of Cohesion) metric 227 Less memory usages versus speed field, Ada 62 Level menu, graphical views 268 libraries, standard Ada, directory for 62 including in Analysis Log 105 Pascal, paths for 89 Library directories field, Ada 63 license server connection timeout 328 licensing specifying level from Help menu 14 Licensing option, Help menu 14 line endings for source files 109 line numbers, displaying in Source Editor 166 line termination style for reports 210 for saving source files 110 Lines metric 229

@lisfile.txt file 327
Local menu, graphical views 268
local object declarations, including in database, C++ 69
local parameters, listed in Entity Locator 26
Lucene, Apache syntax 148

#### Μ

Macintosh line termination style 110 Macro Cross-Reference report 217 macros adding in bulk, C++ 73 changing, Ada 64 compiler-specific, C++ 68 defining in command line 327 defining on command line, Ada 64 defining, Ada 63 defining, C++ 72, 79 of editing changes, recording and replaying 180 expansion text for, C++ 69 importing, Ada 64 listing in Filter Area 25 as octagons, in graphical views 263 overriding MSVC project settings for 58 recording references when expanding, C++ 69 reports about 216, 217 undefined, C++ 74 See also objects Macros category Ada 63 C++ 72.79 Fortran 83 Pascal 89 Main subprograms field, Ada 63 makefile 329 Manage Architectures option, Project menu 195, 201 Margins field, Editor options 111 "Mastering Regular Expressions" (O'Reilly) 159 Max Inheritance Tree metric 227 McCabe (Cyclomatic) complexity 222, 229 members cohesion of 227 default 264 displayed in graphical views 268 graphical views of 258 private 269 protected 269

public 269 Members menu, graphical views 268 memory caching include files, C++ 69 optimizing analysis to use less, Ada 62 menu bar 20 menus. See specific menus merging, in Entity Comparison area 303 methods cohesion of 227 instance 227 listing in Filter Area 25 metrics about 227 metrics 194, 200 calculating automatically 51 configuring 52 displayed in Information Browser 135 exporting to CSV file 51, 231, 235 exporting to HTML 51, 231, 234 for project 232, 233 graphs for 237 in Information Browser 231 list of, online 222, 226, 231, 232, 233 reports about 226, 227, 228, 229, 231 selecting 53 Metrics Browser 231 Metrics category, Project Configuration dialog 52 Metrics Export 195 Metrics Export menu option 201 Metrics menu 334 Browse Metric Charts option 241 Browse Metrics option 231, 233 Configure Metric Charts option 231, 237 Export Metrics option 51, 231, 235 Metrics Summary option 231, 232 Metrics Summary 194, 200 Metrics Summary option, Metrics menu 231, 232 Microsoft Visio files, saving graphical views as 276 Microsoft Visual C++ project. See MSVC project Microsoft Visual C++, as editor 117 minus sign (-) collapsing tree in Information Browser 132 Modified metric 229 modules, listing in Filter Area 25 MSVC project, importing into Understand project 37, 57 multiple users, initialization files for 22

#### Ν

Name menu, graphical views 268 named root portability mode 47 named roots 107 Namespaces category, Pascal 89 Navigation category, Editor options 116 Navigation Mode, Source Editor 116 nested comments, C++ 68 New Architecture option, Project menu 202, 203 New File option, File menu 181 -new option, understand command 328 New Project option, File menu 35 New Project Wizard 35, 106 next button 18 Next icon 29 NIM (Count of Instance Methods) metric 227 NIV (Count of Instance Variables) metric 227 No Truncation text option 270 No Wrap text option 270 -no\_splashscreen option, understand command 328 NOC (Count of Derived Classes) metric 227 Node.js analysis 94 -noproject option, understand command 328 NTFS filesystem 97

### 0

**Object Cross-Reference report 216** Objective C/C++ 15 configuration 75 enabling 41 object-oriented metrics 227 objects displayed in graphical views 268 listing in Project Window 25 reports about 216, 224 Objects menu, graphical views 268 octagons in graphical views 263 online help 16, 22 Open File option, File menu 181 Open last project at startup field, General options 96 Open Project option, File menu 22 Operators menu, graphical views 268 operators, displayed in graphical views 268 Options option, Tools menu 95 Options subcategory, Reports 55 or operators, including in strict complexity, Ada 62

Output subcategory, Reports 54, 208

#### Ρ

Package View 258 packages declaration structure for 258 listing in Filter Area 25 reports about 221 With By relationships for 258 With relationships for 258 page guide, showing 110 Page Setup option, File menu 190 parallelograms, in graphical views 263 parameter declarations without parentheses, Fortran 82 parameters cross-reference information for, C++ 69 included in listing of 130 relationships between formal and actual, Ada 62 reports about 216, 223, 224 See also objects Parameters menu, graphical views 269 parent of class. See base classes of entity 254 Parent Declaration View 258, 259 Parent Lib Unit View 254 parentheses, matching. See brackets parsing. See analyzing projects Pascal configuration for 88 macro definitions 89 namespaces directory 89 search paths 90 standard libraries, paths for 89 versions supported 15 Pascal category, Project Configuration dialog 88 Patch File area, Change Results window 301, 304 patching lines of context, in Entity Comparison area 303.304 PATH definition 320 path highlighting, in graphical views 251 pattern matching. See regular expressions PC line termination style for reports 210 for saving source files 110 .pcn file 215

PDF, saving graphical views to 279 Percent Lack of Cohesion metric 227 period (.) in regular expressions 158 Perl API for custom reports 32 CodeCheck scripts 293 interface for custom reports 213, 231 scripts 19 Perl API Documentation option, Help menu 16, 32, 231 permission checking 97 PHP files 15.94 PHP version field 94 pin icons. See pushpin icons pink background for text 115 PL/M configuration for 90 versions supported 15 PL/M category, Project Configuration dialog 90 plus sign (+) expanding tree in Information Browser 132 in regular expressions 158 PNG format, saving graphical views as 276 Popup Menu category, Tool Configurations dialog 313 Portability category, Understand Options dialog 107 portability of project 47 pound sign (#) comment in command line file 327 pragma statements, defining macros referenced in, Ada 63 Predeclared entities file field, Pascal 88 Prefix Headers Category, C++ Includes 79 Prepend the names of externally linkable entities with field C++ 69 Fortran 82 Prepend the names of JNI/KNI external entities with field, Java 84 preprocessor directives, C++ 72 Preprocessor field, Ada 61 preprocessor macros. See macros preprocessor statements, green text for 115 preprocessor support, enabling, Fortran 82 previous button 18 Previous icon 29 Print Entity Graph option, File menu 278 Print option, File menu 190 printing CodeCheck results 290

graphical views 278 source files 190. 278 Private Members menu, graphical views 269 procedures in graphical views 269 reports about 215 See also invocation; functions; program units Program Unit Complexity report 222 Program Unit Cross-Reference report 215 Program Unit Metrics report 228 program units metrics about 228 reports about 219, 220, 221, 222, 225, 228 With relationships for 221, 225 programming languages selecting for project 41 setting in Source Editor 168 project 19 adding source files on command line 322 analyzing (parsing). See analyzing projects closing 22 closing automatically when opening new project 100 configuring 39 creating 35 creating, with New Project Wizard 106 directory hierarchy for, displaying 137 existing, opening 21 metrics for 226, 232, 233 opening most-recent project at startup 96 portability of 47 saving configuration of 40 toolbar for 162 Project Browser 137 Project Browser option, View menu 137 Project Configuration dialog 39, 208 Ada category 61 Assembly category 65 C# category 80 C++ (Strict) category 75, 79 C++ category 68 COBOL category 66 File Options category 49 File Types category 48 Files category 42 Fortran category 81 Java category 84 JOVIAL category 86

Languages category 41 Metrics category 52 Pascal category 88 PL/M category 90 Python category 92 Reports category 54 saving configuration 40 Scheduled Activities category 50 Visual Studio category 57 Web category 93 Project Graphs option, Graphs menu 32, 213 Project Interactive Reports option, Reports menu 32, 213 Project menu 333 Analyze All Files option 51, 121 Analyze Changed Files option 51, 121 Browse Architectures option 193 Configure option 39 Manage Architectures option 201 New Architecture option 202, 203 Project Overview Charts option 246 Rescan Project Directories option 46, 51 Rescan Watched Project Directories option 46 Project Metrics report 226 Project Overview Charts option, Project menu 246 Prompt before closing the current project field, User Interface Alerts options 100 Prompt for missing include files field 70, 78 Prompt for missing includes field, C++ Includes 71, 78 Prompt on parse errors field Ada 63 Fortran 83 \$Prompt variables 307, 308, 309 prompt when files modified externally 110 Protected Members menu, graphical views 269 Public Members menu, graphical views 269 pushpin icons 18 Python API for custom reports 32 configuration for 92 versions supported 15 Python API Documentation option, Help menu 16, 32 Python category, Project Configuration dialog 92

#### Q

quality reports 222

question mark (?)
in regular expressions 158
wild card 158
-quiet\_startup option, understand command 328
quote (") prefixing octal constants or string literals 82
quotes ("") surrounding normal includes 71

### R

read-only access 49 Recent files most recently use list field, User Interface Lists options 99 Recent projects most recently use list field, User Interface Lists options 99 Recent Projects option, File menu 22 Record Macro option, Tools menu 180 rectangles, in graphical views 263 rectangles, rounded, in graphical views 263 rectangular area, copy and paste 168 rectangular text selection 168 red project file icon 167 red text 115 refactoring 171 Extract Function 174 Extract Temp 176 Inline Function 173 Inline Temp 175 Rename 172 Reference Count option 197 reference files, C# 80 References category, C# 80 References, in Information Browser 135 regular expressions book about 159 in filters for Entity Locator 158 in Find dialog 163 Reindent Selection command 180 relationship 19 relative portability mode 47 Rename Architecture menu option 195 Rename Declaration View 258 rename declarations displayed in graphical views 269 reports about 221 Rename entity 172 Renames menu, graphical views 269 Renames report 221 Replace in Files area 153

Replace in Files option, Search menu 153 Replacement Text category C++ Includes 72 Fortran Includes 83 PL/M Includes 91 Replay Macro option, Tools menu 181 reports 32, 208 anti-virus software, turning off while generating reports 210 canceling generation of 210 categories of 212 colors in 210 configuring 208 cross-reference reports 214 customizing 32 customizing with Perl or C 213, 231 entity name format in 55 filenames for 211 generating 208, 210 generation time of 55, 210 HTML output 54, 210 HTML output, entity index for 211 list of, choosing from 56, 209 metrics reports 226, 231 quality reports 222 structure reports 219 text output 55, 210 viewing 211 See also specific reports Reports category, Project Configuration dialog 54 **Options subcategory** 55 Output subcategory 54, 208 Selected subcategory 56, 209 Reports menu 334 Configure Reports option 208 Generate Reports option 210 Project Interactive Reports option 32, 213 View Reports option 211 Rescan project before analyzing changed files field 105 Rescan Project Directories option, Project menu 46, 51 Rescan Watched Project Directories option 46 Restore Default Position field 101 RFC (Count of All Methods) metric 227 **Right-click Menu** external tools available in 308, 313 Find in Files option 150 right-clicking 18, 127

+ or - sign in Information Browser tree 132 on Analyze icon 121 anywhere in Understand 23 in Architecture Browser 194 in background of graphical views 263 bold heading in Information Browser 133 creating windows by 23 on entities in graphical views 250 on entities to display source 134 on entities in Entity Locator 26 on entities in Entity Locator entities 157 on entities in Information Browser 24 on entities in Source Editor 23, 29 in Entity Filter 24 in Entity Locator 155 on Entity Locator column headers 156, 158 in Information Browser 136 on Selector window 162 reusing windows by 23, 127 on selected text 168, 179 in Source Editor 169 on Source Editor tabs 183 See also Ctrl+right-click RO (read-only) indicator, in Source Editor 168 Root filters, Entity Filter 131 Routines menu, graphical views 269 routines. See functions; procedures row colors, alternating 103 Run Command option, Tools menu 317 RW (read-write) indicator, in Source Editor 168

#### S

Save all modified editor windows when application loses focus field, General options 96 Save All option, File menu 170 Save comments associated with entities field Ada 62 C 76 C++ 69 Java 84 Save duplicate references field, C++ 69 Save macro expansion text field C++ 69 Save on command field, User Interface Alerts options 100 Save on parse field, User Interface Alerts options 100 Save option, File menu 170 saving edits automatically 96 Scale menu, graphical views 269 Scheduled Activities category, Project Configuration dialog 50 Scheduler option, Tools menu 50, 51 Scientific Toolworks website 15 Scope Information, Contextual Information Sidebar 164 Scope List option, View menu 167 Scope List, Source Editor 167 See also Structure Browser, Source Editor 164 toolbar for 162 scripts CodeCheck 293 Perl 19 Search for include files among project files field, C++ Includes 71.78 Search menu 147, 332 Find Entity option 155 Find in Files option 27, 150 Find option 163 Go to Matching Brace option 178 Instant Search option 148 Replace in Files option 153 Search Paths Category, Pascal 90 searching files 147, 150 graphical views 250 Select All option, Edit menu 168 Select Block menu option 178 Selected subcategory, Reports 56, 209 selecting text 302 Selector area 161 sharp sign (#) comment in command line file 327 shortcut commands 104 Show Edge Labels options 273 Show Page Guide field, Editor options 110 Show standard library files field, Analyze options 105 Show tabs field, User Interface options 98 Show the Getting Started dialog on startup field, General options 96 Show the Splash-Screen on startup field, General options 96 Simple Invocation Tree report 220 Simple With Tree report 221 SlickEdit editor 117 sliding frames 18 -SlowConnect option, understand command 328

Soft Wrap option, View menu 180 "Software Engineering with Ada" (Booch) 258 Sort menu, graphical views 270 Sort Selection command 180 sorting entities 129 Sound beep field 100 Sound beep on analysis completion field, Analyze options 105 Source Editor 29, 166 auto-complete options 112 auto-indent options 113, 114 bookmarks, creating 181 bookmarks, navigating 181 bracket matching in 178 Browse Mode in 168 case, changing in 179 closing files 170 colors in, customizing 115, 166 colors in, default 115 commenting and uncommenting code 179 configuring 109 Contextual Information Sidebar (CIS) in 164 copying to clipboard from 168 creating files 181 displaying by right-clicking on entities 134 displaying from Find Results window 27, 152 external editor replacing 117 folding (hiding blocks in) code 178 hiding inactive lines in 178 history of locations visited in, moving through 161 jumping to specific line number 168 keystrokes in, list of 180 language, setting 168 line numbers displayed in 166 list of specific structures in 164, 167 location of 20 macros of editing changes, recording and replaying 180 moving between windows of 29 opening files 181 printing from 111, 190 read-write (RW) and read-only (RO) indicators 168 right-clicking in 23, 29, 169 right-clicking tabs in 183 saving files 170 Scope List in 167 searching and replacing text in source files 163

searching in source files 147 status bar in 167 status icon in 167 tabs in, controlling behavior of 183 toolbar for 162 source files adding to project 37, 42, 138 adding to project on command line 322 analyzing using projects. See project closing 160, 170 creating 181 declaration structure for 258, 260 deleting from project 44 directories for, adding to project 43 directories for, deleting from project 44 displaying by double-clicking entity 134 displaying by right-clicking on entities 134 displaying from Find Results window 27, 152 editing. See Source Editor encoding for 49, 109 excluding from source list 44 external editor for 117 imported classes in 260 imported, report about 221 include files specified as, assembly 65 include files specified as, C++ 70, 77 include files specified as, Fortran 83 include files specified as, PL/M 91 include hierarchy for 254, 256 line endings for 109 list of, generating from command line 327 listing in Filter Area 25 metrics about 228 moving between windows of 29 opening 181 opening from graphical view 250 overriding settings for 46 as parallelograms, in graphical views 263 portability of 47 printing 190, 278 printing, configuration for 111 read-only access for 49 removing from project 138 saving 170 searching 27, 138, 147, 150 Sources tab, Project Configuration dialog 42 sourcestyles.css file 210

spaces, converting tabs to 110 Spacing menu, graphical views 270 special member functions, C++ 69 splash screen, displaying at startup 96 split Source Editor 179 workspace 160 workspace, toolbar for 162 Sql menu, graphical views 270 SQL, embedded, in Pascal 88 square brackets. See brackets src.jar file 84 src.zip file 84 Standard field, Ada 62 standard libraries Ada, directory for 62 displaying in Analysis Log 105 Pascal, paths for 89 Standard Library Paths category, Pascal 89 standards, CodeCheck 283 Start menu, Understand commands in 21 startup Getting Started dialog displayed on 96 Splash-Screen displayed on 96 static functions, displayed in graphical views 270 Static menu, graphical views 270 status bar, Source Editor 167 status icon, in Source Editor 167 status line 20 strict C/C++ analyzer 41, 75, 79 strict complexity, count and/or operators in, Ada 62 Strict metric 229 Structure Browser, Contextual Information Sidebar 164 structure reports 219 structure views 31, 250, 258 See also declaration views; graphical views Styles category, Editor options 115 subprograms, listing in Filter Area 25 subtraction sign (-) collapsing tree in Information Browser 132 Sum Complexity metrics graph 237 support contact information 15 SVG format, saving graphical views as 276 switches, command line understand command 328 synchronizing Information Browser 134 Sysroot field, C++ Includes 77

"System Design in Ada" (Buhr) 258 system include files, C++ 71

#### Т

tab, automatic 113, 114 Table View of graphs 247 Table View, for metrics graphs 239 tabs for windows, displaying 98 tabs, converting to spaces 110 Task View 258 tasks declaration structure for 258 With relationships for 258 technical support contact information 15 temporary bookmark 182 text comparing 300 copying to clipboard 168 generating reports as 32, 55, 208, 210 selecting 168 size of, in graphical views 269 viewing reports as 211 Text Comparison window 300 text favorite 145 Text menu, graphical views 270 TextPad editor 117 title areas, filenames in 98 title bar 18 Title Formats field, User Interface options 98 title page 55 Toggle Overtype option, Edit menu 180 tool commands, importing and exporting 316 **Tool Configurations dialog** Popup Menu category 313 Toolbar category 315 toolbar external tools available from 315 hiding and displaying icons 162 location of 20 visibility of icons in, controlling 316 Toolbar category, Tool Configurations dialog 315 Tools menu 336 Compare Arbitrary Text option 300 Compare Entities option 299 Compare Files/Folders option 296 Editor Macros option 180

Options option 95 Run Command option 317 Scheduler option 50, 51 User Tools option 306, 308, 314, 316 tools, external, See external tools tooltips 116 Treat system includes as user includes field C++ Includes 71 Tree row indentation field, User Interface options 103 treemap CodeCheck 289 metrics 240 Truncate column field Fortran 82 **JOVIAL 86** Truncate Long text option 270 Truncate Medium text option 270 Truncate Short text option 270 truncation at column Fortran 82 **JOVIAL 86** Turbo Pascal 88 Type Cross-Reference report 216 Type Derived From View 254 Type Tree View 254 types displayed in graphical views 270, 271 as hexagons, in graphical views 263 information about 258, 261 listing in Filter Area 25 reports about 216, 225 types derived from 254 Types menu, graphical views 270 Typetext menu, graphical views 271

#### U

.udb file extension 19, 34 UML Class Diagram 258, 260 UML Sequence Diagram 258 Uncomment Selection menu option 179 und command 319, 320 adding files to project 322 analyzing a project 325 options in latest version, listing 322 undefined macros, C++ 74 Undefines category, C++ Macros 74 Understand

compared to compiler 34 contact information 15 features of 13 multiple users for 22 online help for 16, 22 starting 21 starting from command line 328 windows in 18.20 understand command 328 Understand Options dialog 95 Analyze category 105 Command Window category 106 Configure category 106 Dependency category 108 Editor category 109 General category 96 Graphs category 118 Key Bindings category 104, 180 Portability category 107 User Interface category 98 undocking windows 18 Unicode file handling, in Entity Comparison area 303 Uninitialized Items report 224 Unix line termination style, for reports 210 line termination style, for saving source files 110 unknown entities, displaying in graphical views 271 Unknown menu, graphical views 271 unresolved entities displaying in graphical views 271 Unresolved menu, graphical views 271 unresolved variables, listed in Entity Locator 26 Unused Object report 224 Unused Objects and Functions report 224 Unused Program Unit report 225 Unused Type report 225 Update Information Browser field, Browse Mode Editor options 116 uperl command 19 Use alternating row colors field, User Interface options 103 Use case-insensitive lookup for includes field C++ Includes 71 Use default working directory field, General options 97 Use include cache field, C++ 69 Use preprocessor field, Fortran 82 use statements, reports about 225

Use the New Project Wizard when creating new projects field, Configure options 106 Used By View 254 Usedby Menu, graphical views 271 User Interface category, Understand Options dialog 98 user interface, parts of 20 User Tools option, Tools menu 306, 308, 314, 316 users, multiple, initialization files for 22 Uses Menu, graphical views 271 Uses Not Needed report 225 Uses View 254

#### V

variables displayed in graphical views 271 instance 227 listed in Entity Locator 26 metrics about 227 reports about 216, 223, 224 as rounded rectangles, in graphical views 263 uninitialized, report about 224 unresolved 26 See also objects Variables menu, graphical views 271 .vcp file extension 57 .vcw file extension 57 Version field Ada 61 Fortran 81 Java 84 **JOVIAL 86** Pascal 88 vertical bar () in regular expressions 159 Vertical Non-Crossing layout option 267 VHDL terminology 93 versions supported 15 vi editor 117 View menu 333 Analysis Log option 122 Bookmarks option 162, 182 Browse Mode option 168 Contextual Information option 164 Entity Locator option 26, 155 Favorites option 27, 143 Fold All option 178 Hide Inactive Lines option 178

Information Browser 131 Project Browser option 137 Scope List option 167 Soft Wrap option 180 Window Selector option 161 Zoom option 166, 183 View Reports option, Reports menu 211 View SciTools Blog option, Help menu 16 views. See declaration views; graphical views; hierarchical views violations checking for 283 ignoring 290 Visio files, saving graphical views as 276 -visit option, understand command 329 Visit Source field, Browse Mode Editor options 116 Visual C++, as editor 117 Visual Studio as external editor 117 auto-architecture 193 files, synchronizing with Understand project 37, 57 Visual Studio category, Project Configuration dialog 57

#### W

-wait option, understand command 329 warnings, displaying from Analysis Log window 122 watched directories scanning 46, 51 setting 44, 45 Web category, Project Configuration dialog 93 Web support 15 websites external editor information 117 metrics, list of 222, 226, 231, 232, 233 O'Reilly and Associates 159 Scientific Toolworks 15 white file icon 167 whitespace in Entity Comparison area 303 indicators for 110 Whitespace field, Editor options 110 wild cards, in filters for Entity Locator 158 %WINDIR% environment variable 22 Window menu 160, 336 Close All Document Windows option 160, 170 Close option 160, 170

Entity Graph option 250 Windows option 161 Window Selector option, View menu 161 windows 18.20 animated opening and closing of 98 closing 18 creating with Ctrl+right-click 23, 127 docking and undocking 18 docking layout for 98 filenames in title area of 98 frames in, sliding 18 list of 128 list of open windows 161 organizing 160 reusing for graphical views 253 reusing with right-click 23, 127 tabs for, displaying 98 Windows category, Application Styles options 103 Windows category, User Interface options 101 Windows CE project 57 Windows CE workspace 57 Windows folder 22 Windows line termination style for reports 210 for saving source files 110 Windows option, Window menu 161 With Bys menu, graphical views 271 With Tree report 221 With, WithBy Views 254 Withs menu, graphical views 271 Withs Not Needed report 225 WMC (Count of Methods) metric 227 workbench file 57 working directory, default 97 workspace file 57 Wrap Long text option 270 Wrap Medium text option 270 Wrap Mode field, Advanced Editor options 111 Wrap Short text option 270 wrapping lines display 180 printing 111

#### Х

XML Export 195, 206 XML output 242, 243 XML, sharing architectures using 206

### Υ

yellow background for text 115 yellow project file icon 167

#### Ζ

Zoom option, View menu 166, 183 zooming graphical views 251 source views 166